The Review of Financial Studies

Empirical Asset Pricing via Machine Learning*

Shihao Gu
Booth School of Business, University of Chicago

Bryan Kelly
Yale University, AQR Capital Management, and NBER

Dacheng Xiu
Booth School of Business, University of Chicago

We perform a comparative analysis of machine learning methods for the canonical problem of empirical asset pricing: measuring asset risk premiums. We demonstrate large economic gains to investors using machine learning forecasts, in some cases doubling the performance of leading regression-based strategies from the literature. We identify the best-performing methods (trees and neural networks) and trace their predictive gains to allowing nonlinear predictor interactions missed by other methods. All methods agree on the same set of dominant predictive signals, a set that includes variations on momentum, liquidity, and volatility. (*JEL* C52, C55, C58, G0, G1, G17)

Received September 4, 2018; editorial decision September 22, 2019 by Editor Andrew Karolyi. Authors have furnished an Internet Appendix, which is available on the Oxford University Press Web site next to the link to the final published paper online.

*We benefitted from discussions with Joseph Babcock, Si Chen (discussant), Rob Engle, Andrea Frazzini, Amit Goyal (discussant), Lasse Pedersen, Lin Peng (discussant), Alberto Rossi (discussant), and Guofu Zhou (discussant) and seminar and conference participants at Erasmus School of Economics, NYU, Northwestern, Imperial College, National University of Singapore, UIBE, Nanjing University, Tsinghua PBC School of Finance, Fannie Mae, U.S. Securities and Exchange Commission, City University of Hong Kong, Shenzhen Finance Institute at CUHK, NBER Summer Institute, New Methods for the Cross Section of Returns Conference, Chicago Quantitative Alliance Conference, Norwegian Financial Research Conference, EFA, China International Conference in Finance, 10th World Congress of the Bachelier Finance Society, Financial Engineering and Risk Management International Symposium, Toulouse Financial Econometrics Conference, Chicago Conference on New Aspects of Statistics, Financial Econometrics, and Data Science, Tsinghua Workshop on Big Data and Internet Economics, Q group, IQ-KAP Research Prize Symposium, Wolfe Research, INQUIRE UK, Australasian Finance and Banking Conference, Goldman Sachs Global Alternative Risk Premia Conference, AFA, and Swiss Finance Institute. We gratefully acknowledge the computing support from the Research Computing Center at the University of Chicago. The views and opinions expressed are those of the authors and do not necessarily reflect the views of AQR Capital Management, its affiliates, or its employees; do not constitute an offer, solicitation of an offer, or any advice or recommendation, to purchase any securities or other financial instruments, and may not be construed as such. Supplementary data can be found on *The Review of Financial Studies* web site. Send correspondence to Shihao Gu, University of Chicago, Booth School of Business, 5807 S. Woodlawn Ave., Chicago, IL 60637; telephone: +1(310)869-0675. E-mail: shihaogu@chicagobooth.edu.

The Review of Financial Studies 33 (2020) 2223–2273

doi:10.1093/rfs/hhaa009 Advance Access publication February 26, 2020

In this article, we conduct a comparative analysis of machine learning methods for finance. We do so in the context of perhaps the most widely studied problem in finance, that of measuring equity risk premiums.

Our primary contributions are twofold. First, we provide a new set of benchmarks for the predictive accuracy of machine learning methods in measuring risk premiums of the aggregate market and individual stocks. This accuracy is summarized two ways. The first is a high out-of-sample predictive R^2 relative to preceding literature that is robust across a variety of machine learning specifications. Second, and more importantly, we demonstrate large economic gains to investors using machine learning forecasts. A portfolio strategy that times the S&P 500 with neural network forecasts enjoys an annualized out-of-sample Sharpe ratio of 0.77 versus the 0.51 Sharpe ratio of a buy-and-hold investor. And a value-weighted long-short decile spread strategy that takes positions based on stock-level neural network forecasts earns an annualized out-of-sample Sharpe ratio of 1.35, more than doubling the performance of a leading regression-based strategy from the literature.

Return prediction is economically meaningful. The fundamental goal of asset pricing is to understand the behavior of risk premiums.[1] If expected returns were perfectly observed, we would still need theories to explain their behavior and empirical analysis to test those theories. But risk premiums are notoriously difficult to measure: market efficiency *forces* return variation to be dominated by unforecastable news that obscures risk premiums. Our research highlights gains that can be achieved in prediction and identifies the most informative predictor variables. This helps resolve the problem of risk premium measurement, which then facilitates more reliable investigation into economic mechanisms of asset pricing.

Second, we synthesize the empirical asset pricing literature with the field of machine learning. Relative to traditional empirical methods in asset pricing, machine learning accommodates a far more expansive list of potential predictor variables and richer specifications of functional form. It is this flexibility that allows us to push the frontier of risk premium measurement. Interest in machine learning methods for finance has grown tremendously in both academia and industry. This article provides a comparative overview of machine learning methods applied to the two canonical problems of empirical asset pricing: predicting returns in the cross-section and time series. Our view is that the best way for researchers to understand the usefulness of machine learning in the

[1] Our focus is on measuring conditional expected stock returns in excess of the risk-free rate. Academic finance traditionally refers to this quantity as the "risk premium" because of its close connection with equilibrium compensation for bearing equity investment risk. We use the terms "expected return" and "risk premium" interchangeably. One may be interested in potentially distinguishing between different components of expected returns, such as those due to systematic risk compensation, idiosyncratic risk compensation, or even due to mispricing. For machine learning approaches to this problem, see Gu, Kelly, and Xiu (2019) and Kelly, Pruitt, and Su (2019).

field of asset pricing is to apply and compare the performance of each of its methods in familiar empirical problems.

The definition of "machine learning" is inchoate and is often context specific. We use the term to describe (a) a diverse collection of high-dimensional models for statistical prediction, combined with (b) so-called "regularization" methods for model selection and mitigation of overfit, and (c) efficient algorithms for searching among a vast number of potential model specifications.

The high-dimensional nature of machine learning methods (element (a) of this definition) enhances their flexibility relative to more traditional econometric prediction techniques. This flexibility brings hope of better approximating the unknown and likely complex data generating process underlying equity risk premiums. With enhanced flexibility, however, comes a higher propensity of overfitting the data. Element (b) of our machine learning definition describes refinements in implementation that emphasize stable out-of-sample performance to explicitly guard against overfit. Finally, with many predictors it becomes infeasible to exhaustively traverse and compare all model permutations. Element (c) describes clever machine learning tools designed to approximate an optimal specification with manageable computational cost.

A number of aspects of empirical asset pricing make it a particularly attractive field for analysis with machine learning methods.

First, two main research agendas have monopolized modern empirical asset pricing research. The first seeks to describe and understand differences in expected returns across assets. The second focuses on dynamics of the aggregate market equity risk premium. Measurement of an asset's risk premium is fundamentally a problem of prediction—the risk premium is the conditional expectation of a future realized excess return. Machine learning, whose methods are largely specialized for prediction tasks, is thus ideally suited to the problem of risk premium measurement.

Second, the collection of candidate conditioning variables for the risk premium is large. The profession has accumulated a staggering list of predictors that various researchers have argued possess forecasting power for returns. The number of stock-level predictive characteristics reported in the literature numbers in the hundreds and macroeconomic predictors of the aggregate market number in the dozens.[2] Additionally, predictors are often close cousins and highly correlated. Traditional prediction methods break down when the predictor count approaches the observation count or predictors are highly correlated. With an emphasis on variable selection and dimension reduction techniques, machine learning is well suited for such challenging prediction

[2] Green et al. (2013) count 330 stock-level predictive signals in published or circulated drafts. Harvey, Liu, and Zhu (2016) study 316 "factors," which include firm characteristics and common factors, for describing stock return behavior. They note that this is only a subset of those studied in the literature. Welch and Goyal (2008) analyze nearly twenty predictors for the aggregate market return. In both stock and aggregate return predictions, there presumably exists a much larger set of predictors that were tested but failed to predict returns and were thus never reported.

problems by reducing degrees of freedom and condensing redundant variation among predictors.

Third, further complicating the problem is ambiguity about the functional forms through which the high-dimensional predictor sets enter into risk premiums. Should they enter linearly? If nonlinearities are needed, which form should they take? Must we consider interactions among predictors? Such questions rapidly proliferate the set of potential model specifications. The theoretical literature offers little guidance for winnowing the list of conditioning variables and functional forms. Three aspects of machine learning make it well suited for problems of ambiguous functional form. The first is its diversity. As a suite of dissimilar methods, it casts a wide net in its specification search. Second, with methods ranging from generalized linear models to regression trees and neural networks, machine learning is explicitly designed to approximate complex nonlinear associations. Third, parameter penalization and conservative model selection criteria complement the breadth of functional forms spanned by these methods in order to avoid overfit biases and false discovery.

We study a set of candidate models that are potentially well suited to address the three empirical challenges outlined above. They constitute the canon of methods one would encounter in a graduate level machine learning textbook.[3] This includes linear regression, generalized linear models with penalization, dimension reduction via principal components regression (PCR) and partial least squares (PLS), regression trees (including boosted trees and random forests), and neural networks. This is not an exhaustive analysis of all methods. For example, we exclude support vector machines as these share an equivalence with other methods that we study[4] and are primarily used for classification problems. Nonetheless, our list is designed to be representative of predictive analytics tools from various branches of the machine learning tool kit.

We conduct a large-scale empirical analysis, investigating nearly 30,000 individual stocks over 60 years from 1957 to 2016. Our predictor set includes 94 characteristics for each stock, interactions of each characteristic with eight aggregate time-series variables, and 74 industry sector dummy variables, totaling more than 900 baseline signals. Some of our methods expand this predictor set much further by including nonlinear transformations and interactions of the baseline signals. We establish the following empirical facts about machine learning for return prediction.

At the broadest level, our main empirical finding is that machine learning as a whole has the potential to improve our empirical understanding of asset returns. It digests our predictor data set, which is massive from the perspective of the existing literature, into a return forecasting model that dominates traditional

[3] See, for example, Hastie, Tibshirani, and Friedman (2009).

[4] See, for example, Jaggi (2013) and Hastie, Tibshirani, and Friedman (2009), who discuss the equivalence of support vector machines with the lasso. For an application of the kernel trick to the cross-section of returns, see Kozak (2019).

approaches. The immediate implication is that machine learning aids in solving practical investments problems, such as market timing, portfolio choice, and risk management, justifying its role in the business architecture of the fintech industry.

Consider as a benchmark a panel regression of individual stock returns onto three lagged stock-level characteristics: size, book-to-market, and momentum. This benchmark has a number of attractive features. It is parsimonious and simple, and comparing against this benchmark is conservative because it is highly selected (the characteristics it includes are routinely demonstrated to be among the most robust return predictors). Lewellen (2015) demonstrates that this model performs about as well as larger and more complex stock prediction models studied in the literature.

In our sample, which is longer and wider (more observations in terms of both dates and stocks) than that studied in Lewellen (2015), the out-of-sample R^2 from the benchmark model is 0.16% per month for the panel of individual stock returns. When we expand the ordinary least squares (OLS) panel model to include our set of 900+ predictors, predictability vanishes immediately, as evidenced by the R^2 dropping deeply into negative territory. This is not surprising. With so many parameters to estimate, efficiency of OLS regression deteriorates precipitously and therefore produces forecasts that are highly unstable out of sample. This failure of OLS leads us to our next empirical fact.

Vast predictor sets are viable for linear prediction when either penalization or dimension reduction is used. Our first evidence that the machine learning tool kit aids in return prediction emerges from the fact that the "elastic net," which uses parameter shrinkage and variable selection to limit the regression's degrees of freedom, solves the OLS inefficiency problem. In the 900+ predictor regression, elastic net pulls the out-of-sample R^2 into positive territory at 0.11% per month. Principal components regression (PCR) and partial least squares (PLS), which reduce the dimension of the predictor set to a few linear combinations of predictors, further raise the out-of-sample R^2 to 0.26% and 0.27%, respectively. This is in spite of the presence of many likely "fluke" predictors that contribute pure noise to the large model. In other words, the high-dimensional predictor set in a simple linear specification is at least competitive with the status quo low-dimensional model, as long as overparameterization can be controlled.

Next, we expand the model to accommodate nonlinear predictive relationships via generalized linear models, regression trees, and neural networks. Allowing for nonlinearities substantially improves predictions. We find that trees and neural networks unambiguously improve return prediction with monthly stock-level R^2's between 0.33% and 0.40%. But the generalized linear model, which introduces nonlinearity via spline functions of each individual baseline predictor (but with no predictor interactions), fails to robustly outperform the linear specification. This suggests that allowing for (potentially complex) interactions among the baseline predictors is a crucial aspect of nonlinearities in the expected return function. As part of our analysis,

we discuss why generalized linear models are comparatively poorly suited for capturing predictor interactions.

Shallow learning outperforms deeper learning. When we consider a range of neural networks from very shallow (a single hidden layer) to deeper networks (up to five hidden layers), we find that neural network performance peaks at three hidden layers then declines as more layers are added. Likewise, the boosted tree and random forest algorithms tend to select trees with few "leaves" (on average less than six leaves) in our analysis. This is likely an artifact of the relatively small amount of data and tiny signal-to-noise ratio for our return prediction problem, in comparison to the kinds of nonfinancial settings in which deep learning thrives thanks to astronomical data sets and strong signals (such as computer vision).

The distance between nonlinear methods and the benchmark widens when predicting portfolio returns. We build bottom-up portfolio-level return forecasts from the stock-level forecasts produced by our models. Consider, for example, bottom-up forecasts of the S&P 500 portfolio return. By aggregating stock-level forecasts from the benchmark three-characteristic OLS model, we find a monthly S&P 500 predictive R^2 of -0.22%. The bottom-up S&P 500 forecast from the generalized linear model, in contrast, delivers an R^2 of 0.71%. Trees and neural networks improve upon this further, generating monthly out-of-sample R^2's between 1.08% to 1.80% per month. The same pattern emerges for forecasting a variety of characteristic factor portfolios, such as those formed on the basis of size, value, investment, profitability, and momentum. In particular, a neural network with three layers produces a positive out-of-sample predictive R^2 for *every* factor portfolio we consider.

More pronounced predictive power at the portfolio level versus the stock level is driven by the fact that individual stock returns behave erratically for some of the smallest and least liquid stocks in our sample. Aggregating into portfolios averages out much of the unpredictable stock-level noise and boosts the signal strength, which helps in detecting the predictive gains from machine learning.

The economic gains from machine learning forecasts are large. Our tests show clear *statistical* rejections of the OLS benchmark and other linear models in favor of nonlinear machine learning tools. The evidence for *economic* gains from machine learning forecasts—in the form of portfolio Sharpe ratios—are likewise impressive. For example, an investor who times the S&P 500 based on bottom-up neural network forecasts enjoys a 26-percentage-point increase in annualized out-of-sample Sharpe ratio, to 0.77, relative to the 0.51 Sharpe ratio of a buy-and-hold investor. And when we form a long-short decile spread directly sorted on stock return predictions from a neural network, the strategy earns an annualized out-of-sample Sharpe ratio of 1.35 (value-weighted) and 2.45 (equal-weighted). In contrast, an analogous long-short strategy using forecasts from the benchmark OLS model delivers Sharpe ratios of 0.61 and 0.83, respectively.

The most successful predictors are price trends, liquidity, and volatility. All of the methods we study produce a very similar ranking of the most informative stock-level predictors, which fall into three main categories. First, and most informative of all, are price trend variables including stock momentum, industry momentum, and short-term reversal. Next are liquidity variables including market value, dollar volume, and bid-ask spread. Finally, return volatility, idiosyncratic volatility, market beta, and beta squared are also among the leading predictors in all models we consider.

Simulation offers a better understanding of our machine learning findings. In Internet Appendix A, we perform Monte Carlo simulations that support the above interpretations of our analysis. We apply machine learning to simulated data from two different data generating processes. Both produce data from a high dimensional predictor set. But in one, individual predictors enter only linearly and additively, while in the other predictors can enter through nonlinear transformations and via pairwise interactions. When we apply our machine learning repertoire to the simulated data sets, we find that linear and generalized linear methods dominate in the linear and uninteracted setting, yet tree-based methods and neural networks significantly outperform in the nonlinear and interactive setting.

Machine learning has great potential for improving risk premium *measurement*, which is fundamentally a problem of prediction. It amounts to best approximating the conditional expectation $\mathrm{E}(r_{i,t+1}|\mathcal{F}_t)$, where $r_{i,t+1}$ is an asset's return in excess of the risk-free rate, and $\mathcal{F}_t$ is the true and unobservable information set of market participants. This is a domain in which machine learning algorithms excel.

But these improved predictions are *only* measurements. The measurements do not tell us about economic *mechanisms* or *equilibria*. Machine learning methods on their own do not identify deep fundamental associations among asset prices and conditioning variables. When the objective is to understand economic mechanisms, machine learning still may be useful. It requires the economist to add structure—to build a hypothesized mechanism into the estimation problem—and decide how to introduce a machine learning algorithm subject to this structure. A nascent literature is marrying machine learning to equilibrium asset pricing (e.g., Kelly, Pruitt, and Su 2019; Gu, Kelly, and Xiu 2019; Feng, Giglio, and Xiu forthcoming), and this remains an exciting direction for future research.

Our work extends the empirical literature on stock return prediction, which comes in two basic strands. The first strand models differences in expected returns across stocks as a function of stock-level characteristics, and is exemplified by Fama and French (2008) and Lewellen (2015). The typical approach in this literature runs cross-sectional regressions[5] of future stock

[5] In addition to using a least squares regression, the literature often sorts assets into portfolios on the basis of characteristics and studies portfolio averages, a form of nonparametric regression.

returns on a few lagged stock characteristics. The second strand forecasts the time series of returns and is surveyed by Welch and Goyal (2008), Koijen and Nieuwerburgh (2011), and Rapach and Zhou (2013). This literature typically conducts time-series regressions of broad aggregate portfolio returns on a small number of macroeconomic predictor variables.

These traditional methods have potentially severe limitations that more advanced statistical tools in machine learning can help overcome. Most important is that regressions and portfolio sorts are ill-suited to handle the large numbers of predictor variables that the literature has accumulated over five decades. The challenge is how to assess the incremental predictive content of a newly proposed predictor while jointly controlling for the gamut of extant signals (or, relatedly, handling the multiple comparisons and false discovery problem). Our primary contribution is to demonstrate potent return predictability that is harnessable from the large collection of existing variables when machine learning methods are used.

Machine learning methods have sporadically appeared in the asset pricing literature. Rapach, Strauss, and Zhou (2013) apply lasso to predict global equity market returns using lagged returns of all countries. Several papers apply neural networks to forecast derivatives prices (Hutchinson, Lo, and Poggio 1994; Yao, Li, and Tan 2000, among others). Khandani, Kim, and Lo (2010) and Butaru et al. (2016) use regression trees to predict consumer credit card delinquencies and defaults. Sirignano, Sadhwani, and Giesecke (2016) estimate a deep neural network for mortgage prepayment, delinquency, and foreclosure. Heaton, Polson, and Witte (2016) develop a neural network for portfolio selection.

Recently, variations of machine learning methods have been used to study the cross-section of stock returns. Harvey and Liu (2016) study the multiple comparisons problem using a bootstrap procedure. Giglio and Xiu (2016) and Kelly, Pruitt, and Su (2019) use dimension reduction methods to estimate and test factor pricing models. Moritz and Zimmermann (2016) apply tree-based models to portfolio sorting. Kozak, Nagel, and Santosh (2020) and Freyberger, Neuhierl, and Weber (2020) use shrinkage and selection methods to, respectively, approximate a stochastic discount factor and a nonlinear function for expected returns. The focus of our paper is to simultaneously explore a wide range of machine learning methods to study the behavior of expected stock returns, with a particular emphasis on comparative analysis among methods.

1. Methodology

This section describes the collection of machine learning methods that we use in our analysis. In each subsection we introduce a new method and describe it in terms of its three fundamental elements. First is the statistical model describing a method's general functional form for risk premium predictions. The second is an objective function for estimating model parameters. All of

our estimates share the basic objective of minimizing mean squared predictions error (MSE). Regularization is introduced through variations on the MSE objective, such as adding parameterization penalties and robustification against outliers. These modifications are designed to avoid problems with overfit and improve models' out-of-sample predictive performance. Finally, even with a small number of predictors, the set of model permutations expands rapidly when one considers nonlinear predictor transformations. This proliferation is compounded in our already high dimension predictor set. The third element in each subsection describes computational algorithms for efficiently identifying the optimal specification among the permutations encompassed by a given method.

As we present each method, we aim to provide a sufficiently in-depth description of the *statistical model* so that a reader having no machine learning background can understand the basic model structure without needing to consult outside sources. At the same time, when discussing the *computational methods* for estimating each model, we are deliberately terse. There are many variants of each algorithm, and each has its own subtle technical nuances. To avoid bogging down the reader with programming details, we describe our specific implementation choices in Internet Appendix B and refer readers to original sources for further background. Internet Appendix C summarizes the literature on statistical properties of each estimator.

In its most general form, we describe an asset's excess return as an additive prediction error model:

$$r_{i,t+1} = \mathrm{E}_t(r_{i,t+1}) + \epsilon_{i,t+1}, \tag{1}$$

where

$$\mathrm{E}_t(r_{i,t+1}) = g^\star(z_{i,t}). \tag{2}$$

Stocks are indexed as $i = 1, \ldots, N_t$ and months by $t = 1, \ldots, T$. For ease of presentation, we assume a balanced panel of stocks, and defer the discussion on missing data to Section 2.1. Our objective is to isolate a representation of $\mathrm{E}_t(r_{i,t+1})$ as a function of predictor variables that maximizes the out-of-sample explanatory power for realized $r_{i,t+1}$. We denote those predictors as the P-dimensional vector $z_{i,t}$, and assume the conditional expected return $g^\star(\cdot)$ is a flexible function of these predictors.

Despite its flexibility, this framework imposes some important restrictions. The $g^\star(\cdot)$ function depends neither on i nor t. By maintaining the same form over time and across different stocks, the model leverages information from the entire panel and lends stability to estimates of risk premiums for any individual asset. This contrasts with standard asset pricing approaches that reestimate a cross-sectional model each time period, or that independently estimate time-series models for each stock. Also, $g^\star(\cdot)$ depends on z only through $z_{i,t}$. This means our prediction does not use information from the history prior to t, or from individual stocks other than the ith.

1.1 Sample splitting and tuning via validation

Important preliminary steps (prior to discussing specific models and regularization approaches) are to understand how we design disjoint subsamples for estimation and testing and to introduce the notion of "hyperparameter tuning."

The regularization procedures discussed below, which are machine learning's primary defense against overfitting, rely on a choice of hyperparameters (or, synonymously, "tuning parameters"). These are critical to the performance of machine learning methods as they control model complexity. Hyperparameters include, for example, the penalization parameters in lasso and elastic net, the number of iterated trees in boosting, the number of random trees in a forest, and the depth of the trees. In most cases, there is little theoretical guidance for how to "tune" hyperparameters for optimized out-of-sample performance.[6]

We follow the most common approach in the literature and select tuning parameters adaptively from the data in a validation sample. In particular, we divide our sample into three disjoint time periods that maintain the temporal ordering of the data. The first, or "training," subsample is used to estimate the model subject to a specific set of tuning parameter values.

The second, or "validation," sample is used for tuning the hyperparameters. We construct forecasts for data points in the validation sample based on the estimated model from the training sample. Next, we calculate the objective function based on forecast errors from the validation sample, and iteratively search for hyperparameters that optimize the validation objective (at each step reestimating the model from the training data subject to the prevailing hyperparameter values).

Tuning parameters are chosen from the validation sample taking into account estimated parameters, but the parameters are estimated from the training data alone. The idea of validation is to simulate an out-of-sample test of the model. Hyperparameter tuning amounts to searching for a degree of model complexity that tends to produce reliable out-of-sample performance. The validation sample fits are of course not truly out of sample, because they are used for tuning, which is in turn an input to the estimation. Thus, the third, or "testing," subsample, which is used for neither estimation nor tuning, is truly out of sample and thus is used to evaluate a method's predictive performance. Internet Appendix D provides further details for our sample splitting scheme, and Internet Appendix E summarizes the hyperparameter tuning schemes for each model.

1.2 Simple linear

We begin our model description with the least complex method in our analysis, the simple linear predictive regression model estimated via ordinary least squares (OLS). Although we expect this to perform poorly in our high

[6] In machine learning, a "hyperparameter" governs the extent of estimator regularization. This usage is related to, but different from, its meaning in Bayesian statistics as a parameter of a prior distribution.

dimension problem, we use it as a reference point for emphasizing the distinctive features of more sophisticated methods.

The simple linear model imposes that conditional expectations $g^\star(\cdot)$ can be approximated by a linear function of the raw predictor variables and the parameter vector, θ,

$$g(z_{i,t};\theta)=z_{i,t}'\theta. \tag{3}$$

This model imposes a simple regression specification and does not allow for nonlinear effects or interactions between predictors.

Our baseline estimation of the simple linear model uses a standard least squares, or "l_2," objective function:

$$\mathcal{L}(\theta)=\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(r_{i,t+1}-g(z_{i,t};\theta)\right)^2. \tag{4}$$

Minimizing $\mathcal{L}(\theta)$ yields the pooled OLS estimator. The convenience of the baseline l_2 objective function is that it offers analytical estimates and thus avoids sophisticated optimization and computation.

1.2.1 Extension: Robust objective functions. In some cases, replacing Equation (4) with a weighted least squares objective, such as

$$\mathcal{L}_W(\theta)=\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}w_{i,t}\left(r_{i,t+1}-g(z_{i,t};\theta)\right)^2, \tag{5}$$

can possibly improve the predictive performance. This weighted least squares objective allows the econometrician to tilt estimates toward observations that are more statistically or economically informative. For example, one variation that we consider sets $w_{i,t}$ inversely proportional to the number of stocks at time t. This imposes that every month has the same contribution to the model regardless of how many stocks are available that month. This also amounts to equally weighting the squared loss of all stocks available at time t. Another variation that we consider sets $w_{i,t}$ proportional to the equity market value of stock i at time t. This value weighted loss function underweights small stocks in favor of large stocks, and is motivated by the economic rational that small stocks represent a large fraction of the traded universe by count while constituting a tiny fraction of aggregate market capitalization.[7]

Heavy tails are a well-known attribute of financial returns and stock-level predictor variables. Convexity of the least squares objective (4) places extreme emphasis on large errors, thus outliers can undermine the stability of OLS-based

[7] As of Fama and French (2008), the smallest 20% of stocks compose only 3% of aggregate market capitalization. An example of a statistically motivated weighting scheme uses $w_{i,t}$ inversely proportional to an observation's estimated error variance, a choice that potentially improves prediction efficiency in the spirit of generalized least squares.

predictions. The statistics literature, long aware of this problem, has developed modified least squares objective functions that tend to produce more stable forecasts than OLS in the presence of extreme observations.[8] In the machine learning literature, a common choice for counteracting the deleterious effect of heavy-tailed observations is the Huber robust objective function, which is defined as

$$\mathcal{L}_H(\theta)=\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}H\left(r_{i,t+1}-g(z_{i,t};\theta),\xi\right), \tag{6}$$

where

$$H(x;\xi)=\begin{cases} x^2, & \text{if} \quad |x|\leq\xi; \\ 2\xi|x|-\xi^2, & \text{if} \quad |x|>\xi. \end{cases}.$$

The Huber loss, $H(\cdot)$, is a hybrid of squared loss for relatively small errors and absolute loss for relatively large errors, where the combination is controlled by a tuning parameter, ξ, that can be optimized adaptively from the data.[9]

Although this detour introduces robust objective functions in the context of the simple linear model, they are easily applicable in almost all the methods we study. In our empirical analysis, we study the predictive benefits of robust loss functions in multiple machine learning methods.

1.3 Penalized linear

The simple linear model is bound to fail in the presence of many predictors. When the number of predictors P approaches the number of observations T, the linear model becomes inefficient or even inconsistent.[10] It begins to overfit noise rather than extracting signal. This is particularly troublesome for the problem of return prediction where the signal-to-noise ratio is notoriously low.

Crucial for avoiding overfit is reducing the number of estimated parameters. The most common machine learning device for imposing parameter parsimony is to append a penalty to the objective function in order to favor more parsimonious specifications. This "regularization" of the estimation problem mechanically deteriorates a model's in-sample performance in hopes that it improves its stability out of sample. This will be the case when penalization manages to reduce the model's fit of noise, while preserving its signal fit.

The statistical model for our penalized linear model is the same as the simple linear model in Equation (3). That is, it continues to consider only the baseline,

[8] Classical analyses include Box (1953), Tukey (1960), and Huber (1964).

[9] OLS is a special case of the (6) with $\xi=\infty$. Although most theoretical analysis from high-dimensional statistics assume that data have sub-Gaussian or subexponential tails, Fan et al. (2017) provide a theoretical justification of using this loss function in the high-dimensional setting as well as a procedure to determine the hyperparameter.

[10] We deliberately compare P with T, instead of with NT, because stock returns share strong cross-sectional dependence, limiting the incremental information contained in new cross-section observations.

untransformed predictors. Penalized methods differ by appending a penalty to the original loss function:

$$\mathcal{L}(\theta;\cdot)=\mathcal{L}(\theta)+\phi(\theta;\cdot). \tag{7}$$

There are several choices for the penalty function $\phi(\theta;\cdot)$. We focus on the popular "elastic net" penalty, which takes the form

$$\phi(\theta;\lambda,\rho)=\lambda(1-\rho)\sum_{j=1}^{P}|\theta_j|+\frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2. \tag{8}$$

The elastic net involves two nonnegative hyperparameters, λ and ρ, and includes two well-known regularizers as special cases. The $\rho=0$ case corresponds to the lasso and uses an absolute value, or "l_1," parameter penalization. The fortunate geometry of the lasso sets coefficients on a subset of covariates to exactly zero. In this sense, the lasso imposes sparsity on the specification and can thus be thought of as a variable *selection* method. The $\rho=1$ case corresponds to ridge regression, which uses an l_2 parameter penalization, that draws all coefficient estimates closer to zero but does not impose exact zeros anywhere. In this sense, ridge is a *shrinkage* method that helps prevent coefficients from becoming unduly large in magnitude. For intermediate values of ρ, the elastic net encourages simple models through both shrinkage and selection.

We adaptively optimize the tuning parameters, λ and ρ, using the validation sample. Our implementation of penalized regression uses the accelerated proximal gradient algorithm and accommodates both least squares and Huber objective functions (see Internet Appendix B.1 for more details).

1.4 Dimension reduction: PCR and PLS

Penalized linear models use shrinkage and variable selection to manage high dimensionality by forcing the coefficients on most regressors near or exactly to zero. This can produce suboptimal forecasts when predictors are highly correlated. A simple example of this problem is a case in which all of the predictors are equal to the forecast target plus an iid noise term. In this situation, choosing a subset of predictors via lasso penalty is inferior to taking a simple average of the predictors and using this as the sole predictor in a univariate regression.

The idea of predictor averaging, as opposed to predictor selection, is the essence of dimension reduction. Forming linear combinations of predictors helps reduce noise to better isolate the signal in predictors and helps decorrelate otherwise highly dependent predictors. Two classic dimension reduction techniques are principal components regression (PCR) and partial least squares (PLS).

PCR consists of a two-step procedure. In the first step, principal components analysis (PCA) combines regressors into a small set of linear combinations that best preserve the covariance structure among the predictors. In the second

step, a few leading components are used in standard predictive regression. That is, PCR regularizes the prediction problem by zeroing out coefficients on low variance components.

A drawback of PCR is that it fails to incorporate the ultimate statistical objective—forecasting returns—in the dimension reduction step. PCA condenses data into components based on the covariation *among* the predictors. This happens prior to the forecasting step and without consideration of how predictors associate with future returns.

In contrast, partial least squares performs dimension reduction by directly exploiting covariation of predictors with the forecast target.[11] PLS regression proceeds as follows. For each predictor j, estimate its univariate return prediction coefficient via OLS. This coefficient, denoted φ_j, reflects the "partial" sensitivity of returns to each predictor j. Next, average all predictors into a single aggregate component with weights proportional to φ_j, placing the highest weight on the strongest univariate predictors, and the least weight on the weakest. In this way, PLS performs its dimension reduction with the ultimate forecasting objective in mind. To form more than one predictive component, the target and all predictors are orthogonalized with respect to previously constructed components, and the above procedure is repeated on the orthogonalized data set. This is iterated until the desired number of PLS components is reached.

Our implementation of PCR and PLS begins from the vectorized version of the linear model in Equations (1)–(3). In particular, we reorganize the linear regression $r_{i,t+1} = z_{i,t}'\theta + \epsilon_{i,t+1}$ as

$$R = Z\theta + E, \tag{9}$$

where R is the $NT \times 1$ vector of $r_{i,t+1}$, Z is the $NT \times P$ matrix of stacked predictors $z_{i,t}$, and E is a $NT \times 1$ vector of residuals $\epsilon_{i,t+1}$.

PCR and PLS take the same general approach to reducing the dimensionality. They both condense the set of predictors from dimension P to a much smaller number of K linear combinations of predictors. Thus, the forecasting model for both methods is written as

$$R = (Z\Omega_K)\theta_K + \tilde{E}. \tag{10}$$

Ω_K is $P \times K$ matrix with columns $w_1, w_2, \ldots, w_K$. Each w_j is the set of linear combination weights used to create the jth predictive components, thus $Z\Omega_K$ is the dimension-reduced version of the original predictor set. Likewise, the predictive coefficient θ_K is now a $K \times 1$ vector rather than $P \times 1$.

[11] See Kelly and Pruitt (2013, 2015) for asymptotic theory of PLS regression and its application to forecasting risk premiums in financial markets.

PCR chooses the combination weights Ω_K recursively. The j^{th} linear combination solves[12]

$$w_j = \arg\max_w \text{Var}(Zw), \quad \text{s.t. } w'w = 1, \quad \text{Cov}(Zw, Zw_l) = 0, \ l = 1, 2, \ldots, j-1. \tag{11}$$

Intuitively, PCR seeks the K linear combinations of Z that most faithfully mimic the full predictor set. The objective illustrates that the choice of components is not based on the forecasting objective at all. Instead, the emphasis of PCR is on finding components that retain the most possible common variation within the predictor set. The well known solution for (11) computes Ω_K via singular value decomposition of Z, and therefore the PCR algorithm is extremely efficient from a computational standpoint.

In contrast to PCR, the PLS objective seeks K linear combinations of Z that have maximal predictive association with the forecast target. The weights used to construct the jth PLS component solve

$$w_j = \arg\max_w \text{Cov}^2(R, Zw), \quad \text{s.t.} \quad w'w = 1, \quad \text{Cov}(Zw, Zw_l) = 0, \quad l = 1, 2, \ldots, j-1. \tag{12}$$

This objective highlights the main distinction between PCR and PLS. PLS is willing to sacrifice how accurately $Z\Omega_K$ approximates Z in order to find components with more potent return predictability. The problem in (12) can be efficiently solved using a number of similar routines, the most prominent being the SIMPLS algorithm of de Jong (1993).

Finally, given a solution for Ω_K, θ_K is estimated in both PCR and PLS via OLS regression of R on $Z\Omega_K$. For both models, K is a hyperparameter that can be determined adaptively from the validation sample.

1.5 Generalized linear

Linear models are popular in practice, in part because they can be thought of as a first-order approximation to the data generating process.[13] When the "true" model is complex and nonlinear, restricting the functional form to be linear introduces approximation error due to model misspecification. Let $g^\star(z_{i,t})$ denote the true model and $g(z_{i,t};\theta)$ the functional form specified by the econometrician. And let $g(z_{i,t};\widehat{\theta})$ and $\widehat{r}_{i,t+1}$ denote the fitted model and its ensuing return forecast. We can decompose a model's forecast error as:

$$r_{i,t+1} - \widehat{r}_{i,t+1} = \underbrace{g^\star(z_{i,t}) - g(z_{i,t};\theta)}_{\text{approximation error}} + \underbrace{g(z_{i,t};\theta) - g(z_{i,t};\widehat{\theta})}_{\text{estimation error}} + \underbrace{\epsilon_{i,t+1}}_{\text{intrinsic error}}.$$

Intrinsic error is irreducible; it is the genuinely unpredictable component of returns associated with news arrival and other sources of randomness in

[12] For two vectors a and b, we denote $\text{Cov}(a,b) := (a-\bar{a})^\intercal(b-\bar{b})$, where $\bar{a}$ is the average of all entries of a. Naturally, we define $\text{Var}(a) := \text{Cov}(a,a)$.

[13] See White (1980) for a discussion of the limitations of linear models as first-order approximations.

financial markets. Estimation error, which arises due to sampling variation, is determined by the data. It is potentially reducible by adding new observations, though this may not be under the econometrician's control. Approximation error is directly controlled by the econometrician and is potentially reducible by incorporating more flexible specifications that improve the model's ability to approximate the true model. But additional flexibility raises the risk of overfitting and destabilizing the model out of sample. In this and the following subsections, we introduce nonparametric models of $g(\cdot)$ with increasing degrees of flexibility, each complemented by regularization methods to mitigate overfit.

The first and most straightforward nonparametric approach that we consider is the generalized linear model. It introduces nonlinear transformations of the original predictors as new additive terms in an otherwise linear model. Generalized linear models are thus the closest nonlinear counterparts to the linear approaches in Sections 1.2 and 1.3.

The model we study adapts the simple linear form by adding a K-term spline series expansion of the predictors

$$g(z;\theta, p(\cdot)) = \sum_{j=1}^{P} p(z_j)'\theta_j, \tag{13}$$

where $p(\cdot) = (p_1(\cdot), p_2(\cdot), \ldots, p_K(\cdot))'$ is a vector of basis functions, and the parameters are now a $K \times N$ matrix $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$. There are many potential choices for spline functions. We adopt a spline series of order two: $\left(1, z, (z-c_1)^2, (z-c_2)^2, \ldots, (z-c_{K-2})^2\right)$, where $c_1, c_2, \ldots c_{K-2}$ are knots.

Because higher-order terms enter additively, forecasting with the generalized linear model can be approached with the same estimation tools as in Section 1.2. In particular, our analysis uses a least squares objective function, both with and without the Huber robustness modification. Because series expansion quickly multiplies the number of model parameters, we use penalization to control degrees of freedom. Our choice of penalization function is specialized for the spline expansion setting and is known as the group lasso. It takes the form

$$\phi(\theta;\lambda, K) = \lambda \sum_{j=1}^{P} \left(\sum_{k=1}^{K} \theta_{j,k}^2 \right)^{1/2}. \tag{14}$$

As its name suggests, the group lasso selects either all K spline terms associated with a given characteristic or none of them. We embed this penalty in the general objective of Equation (7). Group lasso accommodates either least squares or robust Huber objective, and it uses the same accelerated proximal gradient descent as the elastic net. It has two tuning parameters, λ and K.[14]

[14] For additional details, see Internet Appendix B.1. Freyberger, Neuhierl, and Weber (2020) offers a similar model in the return prediction context.

1.6 Boosted regression trees and random forests

The model in (13) captures individual predictors' nonlinear impact on expected returns, but does not account for interactions among predictors. One way to add interactions is to expand the generalized model to include multivariate functions of predictors. While expanding univariate predictors with K basis functions multiplies the number of parameters by a factor of K, multiway interactions increase the parameterization combinatorially. Without a priori assumptions for which interactions to include, the generalized linear model becomes computationally infeasible.[15]

As an alternative, regression trees have become a popular machine learning approach for incorporating multiway predictor interactions. Unlike linear models, trees are fully nonparametric and possess a logic that departs markedly from traditional regressions. At a basic level, trees are designed to find groups of observations that behave similarly to each. A tree "grows" in a sequence of steps. At each step, a new "branch" sorts the data leftover from the preceding step into bins based on one of the predictor variables. This sequential branching slices the space of predictors into rectangular partitions, and approximates the unknown function $g^\star(\cdot)$ with the average value of the outcome variable within each partition.

Figure 1 shows an example with two predictors, "size" and "b/m." The left panel describes how the tree assigns each observation to a partition based on its predictor values. First, observations are sorted on size. Those above the breakpoint of 0.5 are assigned to Category 3. Those with small size are then further sorted by b/m. Observations with small size and b/m below 0.3 are assigned to Category 1, while those with b/m above 0.3 go into Category 2. Finally, forecasts for observations in each partition are defined as the simple average of the outcome variable's value among observations in that partition.

More formally, the prediction of a tree, $\mathcal{T}$, with K "leaves" (terminal nodes), and depth L, can be written as

$$g(z_{i,t};\theta,K,L)=\sum_{k=1}^{K}\theta_k \mathbf{1}_{\{z_{i,t}\in C_k(L)\}}, \tag{15}$$

where $C_k(L)$ is one of the K partitions of the data. Each partition is a product of up to L indicator functions of the predictors. The constant associated with partition k (denoted θ_k) is defined to be the sample average of outcomes within the partition.[16] The example in Figure 1 has the following prediction equation:

$$g(z_{i,t};\theta,3,2)=\theta_1 \mathbf{1}_{\{\text{size}_{i,t}<0.5\}}\mathbf{1}_{\{\text{b/m}_{i,t}<0.3\}}+\theta_2 \mathbf{1}_{\{\text{size}_{i,t}<0.5\}}\mathbf{1}_{\{\text{b/m}_{i,t}\geq 0.3\}}+\theta_3 \mathbf{1}_{\{\text{size}_{i,t}\geq 0.5\}}.$$

[15] Parameter penalization does not solve the difficulty of estimating linear models when the number of predictors is exponentially larger than the number of observations. Instead, one must turn to heuristic optimization algorithms, such as stepwise regression (sequentially adding/dropping variables until some stopping rule is satisfied), variable screening (retaining predictors whose univariate correlations with the prediction target exceed a certain value), or others.

[16] We focus on recursive binary trees for their relative simplicity. Breiman et al. (1984) discuss more complex tree structures.

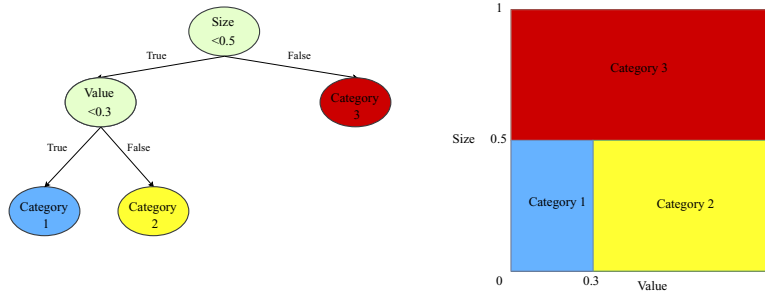


Figure 1
Regression tree example
This figure presents the diagrams of a regression tree (left) and its equivalent representation (right) in the space of two characteristics (size and value). The terminal nodes of the tree are colored in blue, yellow, and red. Based on their values of these two characteristics, the sample of individual stocks is divided into three categories.

To grow a tree is to find bins that best discriminate among the potential outcomes. The specific predictor variable upon which a branch is based, and the specific value where the branch is split, is chosen to minimize forecast error. The expanse of potential tree structures, however, precludes exact optimization. The literature has developed a set of sophisticated optimization heuristics to quickly converge on approximately optimal trees. We follow the algorithm of Breiman et al. (1984) and provide a detailed description of it in Internet Appendix B.2. The basic idea is to myopically optimize forecast error at the start of each branch. At each new level, we choose a sorting variable from the set of predictors and the split value to maximize the discrepancy among average outcomes in each bin.[17] The loss associated with the forecast error for a branch C is often called "impurity," which describes how similarly observations behave on either side of the split. We choose the most popular l_2 impurity for each branch of the tree:

$$H(\theta, C) = \frac{1}{|C|} \sum_{z_{i,t} \in C} (r_{i,t+1} - \theta)^2, \tag{16}$$

where $|C|$ denotes the number of observations in set C. Given C, it is clear that the optimal choice of θ: $\theta = \frac{1}{|C|} \sum_{z_{i,t} \in C} r_{i,t+1}$. The procedure is equivalent to finding the branch C that locally minimizes the impurity. Branching halts when the number of leaves or the depth of the tree reach a prespecified threshold that can be selected adaptively using a validation sample.

Advantages of the tree model are that it is invariant to monotonic transformations of predictors, that it naturally accommodates categorical and numerical data in the same model, that it can approximate potentially severe

[17] Because splits are chosen without consideration of future potential branches, it is possible to myopically bypass an inferior branch that would have led to a future branch with an ultimately superior reduction in forecast error.

nonlinearities, and that a tree of depth L can capture $(L-1)$-way interactions. Their flexibility is also their limitation. Trees are among the prediction methods most prone to overfit, and therefore must be heavily regularized. In our analysis, we consider two "ensemble" tree regularizers that combine forecasts from many different trees into a single forecast.[18]

The first regularization method is "boosting," which recursively combines forecasts from many oversimplified trees.[19] Shallow trees on their own are "weak learners" with minuscule predictive power. The theory behind boosting suggests that many weak learners may, as an ensemble, comprise a single "strong learner" with greater stability than a single complex tree.

The details of our boosting procedure, typically referred to as gradient boosted regression trees (GBRT), are described in Algorithm 4 of Internet Appendix B.2. It starts by fitting a shallow tree (e.g., with depth $L=1$). This oversimplified tree is sure to be a weak predictor with large bias in the training sample. Next, a second simple tree (with the same shallow depth L) is used to fit the prediction residuals from the first tree. Forecasts from these two trees are added together to form an ensemble prediction of the outcome, but the forecast component from the second tree is shrunken by a factor $\nu \in (0,1)$ to help prevent the model from overfitting the residuals. At each new step b, a shallow tree is fitted to the residuals from the model with $b-1$ trees, and its residual forecast is added to the total with a shrinkage weight of ν. This is iterated until there are a total of B trees in the ensemble. The final output is therefore an additive model of shallow trees with three tuning parameters (L, ν, B), which we adaptively choose in the validation step.

Like boosting, a random forest is an ensemble method that combines forecasts from many different trees. It is a variation on a more general procedure known as bootstrap aggregation, or "bagging" (Breiman 2001). The baseline tree bagging procedure draws B different bootstrap samples of the data, fits a separate regression tree to each, then averages their forecasts. Trees for individual bootstrap samples tend to be deep and overfit, making their individual predictions inefficiently variable. Averaging over multiple predictions reduces this variation, thus stabilizing the trees' predictive performance.

Random forests use a variation on bagging designed to reduce the correlation among trees in different bootstrap samples. If, for example, firm size is the dominant return predictor in the data, then most of the bagged trees will have low-level splits on size resulting in substantial correlation among their ultimate predictions. The forest method decorrelates trees using a method known as

[18] The literature also considers a number of other approaches to tree regularization, such as early stopping and post-pruning, both of which are designed to reduce overfit in a single large tree. Ensemble methods demonstrate more reliable performance and are scalable for very large data sets, leading to their increased popularity in recent literature.

[19] Boosting is originally described in Schapire (1990) and Freund (1995) for classification problems to improve the performance of a set of weak learners. Friedman et al. (2000) and Friedman (2001) extend boosting to contexts beyond classification, eventually leading to the gradient boosted regression tree.

"dropout," which considers only a randomly drawn subset of predictors for splitting at each potential branch. Doing so ensures that, in the example, early branches for at least a few trees will split on characteristics other than firm size. This lowers the average correlation among predictions to further improve the variance reduction relative to standard bagging. Depth L of the trees, number of predictors in each split and number of bootstrap samples B are the tuning parameters optimized via validation. Algorithm 3 of the Internet Appendix describes the precise details of our random forest implementation.

1.7 Neural networks

The final nonlinear method that we analyze is the artificial neural network. Arguably the most powerful modeling device in machine learning, neural networks have theoretical underpinnings as "universal approximators" for any smooth predictive association (Hornik, Stinchcombe, and White 1989; Cybenko 1989). They are the currently preferred approach for complex machine learning problems, such as computer vision, natural language processing, and automated game-playing (such as chess and go). Their flexibility draws from the ability to entwine many telescoping layers of nonlinear predictor interactions, earning the synonym "deep learning." At the same time, their complexity ranks neural networks among the least transparent, least interpretable, and most highly parameterized machine learning tools.

Our analysis focuses on traditional "feed-forward" networks. These consist of an "input layer" of raw predictors, one or more "hidden layers" that interact and nonlinearly transform the predictors, and an "output layer" that aggregates hidden layers into an ultimate outcome prediction. Analogous to axons in a biological brain, layers of the networks represent groups of "neurons" with each layer connected by "synapses" that transmit signals among neurons of different layers. Figure 2 shows two illustrative examples.

The number of units in the input layer is equal to the dimension of the predictors, which we set to four in this example (denoted $z_1, ..., z_4$). The left panel shows the simplest possible network that has no hidden layers. Each of the predictor signals is amplified or attenuated according to a 5-dimensional parameter vector, θ, that includes an intercept and one weight parameter per predictor. The output layer aggregates the weighted signals into the forecast $\theta_0 + \sum_{k=1}^{4} z_k \theta_k$; that is, the simplest neural network is a linear regression model.

The model incorporates more flexible predictive associations by adding hidden layers between the inputs and output. The right panel of Figure 2 shows an example with one hidden layer that contains five neurons. Each neuron draws information linearly from all of the input units, just as in the simple network on the left. Then, each neuron applies a nonlinear "activation function" f to its aggregated signal before sending its output to the next layer. For example, the second neuron in the hidden layer transforms inputs into an output as $x_2^{(1)} = f\left(\theta_{2,0}^{(0)} + \sum_{j=1}^{4} z_j \theta_{2,j}^{(0)}\right)$. Lastly, the results from each neuron are linearly

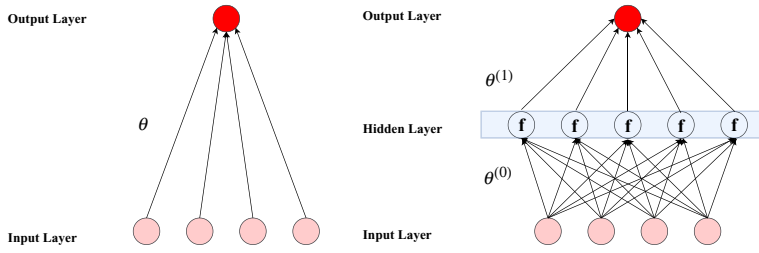


Figure 2
Neural networks
This figure provides diagrams of two simple neural networks with (right) or without (left) a hidden layer. Pink circles denote the input layer, and dark red circles denote the output layer. Each arrow is associated with a weight parameter. In the network with a hidden layer, a nonlinear activation function f transforms the inputs before passing them on to the output.

aggregated into an ultimate output forecast:

$$g(z;\theta)=\theta_0^{(1)}+\sum_{j=1}^{5}x_j^{(1)}\theta_j^{(1)}.$$

Thus, in this example, there are a total of $31=(4+1)\times 5+6$ parameters (five parameters to reach each neuron and six weights to aggregate the neurons into a single output).

One makes many choices when structuring a neural network, including the number of hidden layers, the number of neurons in each layer, and which units are connected. Despite the aforementioned "universal approximation" result that suggests the sufficiency of a single hidden layer, recent literature has shown that deeper networks can often achieve the same accuracy with substantially fewer parameters.[20]

On the other hand, in small data sets simple networks with only a few layers and nodes often perform best. Training a very deep neural network is challenging because it typically involves a large number of parameters, because the objective function is highly nonconvex, and because the recursive calculation of derivatives (known as "back-propagation") is prone to exploding or vanishing gradients.

Selecting a successful network architecture by cross-validation is in general a difficult task. It is unrealistic and unnecessary to find the optimal network by searching over uncountably many architectures. Instead, we fix a variety of network architectures ex ante and estimate each of these. What we hope to achieve is reasonably lower bound on the performance of machine learning methods.

[20] Eldan and Shamir (2016) formally demonstrate that depth—even if increased by one layer—can be exponentially more valuable than increasing width in standard feed-forward neural networks. Ever since the seminal work of Hinton, Osindero, and Teh (2006), the machine learning community has experimented and adopted deeper (and wider) networks, with as many as 152 layers for image recognition (see, e.g., He et al. 2016).

We consider architectures with up to five hidden layers. Our shallowest neural network has a single hidden layer of 32 neurons, which we denoted NN1. Next, NN2 has two hidden layers with 32 and 16 neurons, respectively; NN3 has three hidden layers with 32, 16, and 8 neurons, respectively; NN4 has four hidden layers with 32, 16, 8, and 4 neurons, respectively; and NN5 has five hidden layers with 32, 16, 8, 4, and 2 neurons, respectively. We choose the number of neurons in each layer according to the geometric pyramid rule (see Masters 1993). All architectures are fully connected so each unit receives an input from all units in the layer below. By comparing the performance of NN1 through NN5, we can infer the trade-offs of network depth in the return forecasting problem.[21]

There are many potential choices for the nonlinear activation function (such as sigmoid, hyperbolic, softmax). We use the same activation function at all nodes, and choose a popular functional form in recent literature known as the rectified linear unit (ReLU), defined as[22]

$$\text{ReLU}(x)=\begin{cases}0 & \text{if } x<0\\ x & \text{otherwise,}\end{cases}$$

which encourages sparsity in the number of active neurons and allows for faster derivative evaluation.

Our neural network model has the following general formula. Let $K^{(l)}$ denote the number of neurons in each layer $l=1,...,L$. Define the output of neuron k in layer l as $x_k^{(l)}$. Next, define the vector of outputs for this layer (augmented to include a constant, $x_0^{(l)}$) as $x^{(l)}=(1,x_1^{(l)},...,x_{K^{(l)}}^{(l)})'$. To initialize the network, similarly define the input layer using the raw predictors, $x^{(0)}=(1,z_1,...,z_N)'$. The recursive output formula for the neural network at each neuron in layer $l>0$ is then

$$x_k^{(l)}=\text{ReLU}\left(x^{(l-1)'}\theta_k^{(l-1)}\right), \tag{17}$$

with final output

$$g(z;\theta)=x^{(L-1)'}\theta^{(L-1)}. \tag{18}$$

The number of weight parameters in each hidden layer l is $K^{(l)}(1+K^{(l-1)})$, plus another $1+K^{(L-1)}$ weights for the output layer.

We estimate the neural network weight parameters by minimizing the penalized l_2 objective function of prediction errors. Unlike tree-based algorithms that require "greedy" optimization, training a neural network, in principle, allows for joint updates of all model parameters at each step

[21] We confine the choices of architectures to a small set of five based on our limited sample size (compared to typical neural network applications).

[22] See, for example, Jarrett et al. (2009), Nair and Hinton (2010), and Glorot, Bordes, and Bengio (2011).

of the optimization—a substantial advantage of neural networks over trees. However, the high degree of nonlinearity and nonconvexity in neural networks, together with their rich parameterization, make brute force optimization highly computationally intensive (often to the point of infeasibility). A common solution uses stochastic gradient descent (SGD) to train a neural network. Unlike standard descent that uses the entire training sample to evaluate the gradient at each iteration of the optimization, SGD evaluates the gradient from a small random subset of the data. This approximation sacrifices accuracy for enormous acceleration of the optimization routine.

For the same reasons described above (severe nonlinearity and heavy parameterization), regularization of neural networks requires more care than the methods discussed above. In addition to l_1 penalization of the weight parameters, we simultaneously employ four other regularization techniques in our estimation: learning rate shrinkage, early stopping, batch normalization, and ensembles.

A critical tuning parameter in SGD is the learning rate, which controls the step size of the descent. It is necessary to shrink the learning rate toward zero as the gradient approaches zero, otherwise noise in the calculation of the gradient begins to dominate its directional signal. We adopt the "learning rate shrinkage" algorithm of Kingma and Ba (2014) to adaptively control the learning rate (described further in Algorithm 5 of the Internet Appendix B.3).[23]

Next, "early stopping" is a general machine learning regularization tool. It begins from an initial parameter guess that imposes parsimonious parameterization (e.g., setting all θ values close to zero). In each step of the optimization algorithm, the parameter guesses are gradually updated to reduce prediction errors in the training sample. At each new guess, predictions are also constructed for the validation sample, and the optimization is terminated when the validation sample errors begin to increase. This typically occurs before the prediction errors are minimized in the training sample, hence the name early stopping (see Algorithm 6 of the Internet Appendix B.3). By ending the parameter search early, parameters are shrunken toward the initial guess. It is a popular substitute to l_2 penalization of θ parameters because it achieves regularization at a much lower computational cost.[24] Early stopping can be used alone or together with l_1-regularization, as we do in this paper.

"Batch normalization" (Ioffe and Szegedy 2015) is a simple technique for controlling the variability of predictors across different regions of the network

[23] Relatedly, random subsetting at each SGD iteration adds noise to the optimization procedure, which itself serves as a form of regularization. See Wilson and Martinez (2003).

[24] Early stopping bears a comparatively low computation cost because of only partial optimization, whereas the l_2 regularization, or more generally elastic net, search across tuning parameters fully optimizes the model subject to each tuning parameter guess. As usual, elastic net's l_1-penalty component encourages neurons to connect to limited number of other neurons, whereas its l_2-penalty component shrinks the weight parameters toward zero (a feature known in the neural net literature as "weight decay"). In certain circumstances, early stopping and weight decay are shown to be equivalent. See, for example, Bishop (1995) and Goodfellow, Bengio, and Courville (2016).

and across different data sets. It is motivated by the phenomenon of internal covariate shift in which inputs of hidden layers follow different distributions than their counterparts in the validation sample. This issue is constantly encountered when fitting deep neural networks that involve many parameters and rather complex structures. For each hidden unit in each training step (a "batch"), the algorithm cross-sectionally demeans and variance standardizes the batch inputs to restore the representation power of the unit.

Finally, we adopt an ensemble approach in training our neural networks (see also Hansen and Salamon 1990; Dietterich 2000). In particular, we use multiple random seeds to initialize neural network estimation and construct predictions by averaging forecasts from all networks. This reduces prediction variance because the stochastic nature of the optimization can cause different seeds to produce different forecasts.[25]

1.8 Performance evaluation

To assess predictive performance for individual excess stock return forecasts, we calculate the out-of-sample R^2 as

$$R^2_{\text{oos}} = 1 - \frac{\sum_{(i,t)\in\mathcal{T}_3}(r_{i,t+1} - \widehat{r}_{i,t+1})^2}{\sum_{(i,t)\in\mathcal{T}_3} r^2_{i,t+1}}, \tag{19}$$

where $\mathcal{T}_3$ indicates that fits are only assessed on the testing subsample, whose data never enter into model estimation or tuning. R^2_{oos} pools prediction errors across firms and over time into a grand panel-level assessment of each model.

A subtle but important aspect of our R^2 metric is that the denominator is the sum of squared excess returns *without demeaning*. In many out-of-sample forecasting applications, predictions are compared against historical mean returns. Although this approach is sensible for the aggregate index or long-short portfolios, for example, it is flawed when it comes to analyzing individual stock returns. Predicting future excess stock returns with historical averages typically *underperforms* a naive forecast of zero by a large margin. That is, the historical mean stock return is so noisy that it artificially lowers the bar for "good" forecasting performance. We avoid this pitfall by benchmarking our R^2 against a forecast value of zero. To give an indication of the importance of this choice, when we benchmark model predictions against historical mean stock returns, the out-of-sample monthly R^2 of all methods rises by roughly three percentage points.

To make pairwise comparisons of methods, we use the Diebold and Mariano (1995) test for differences in out-of-sample predictive accuracy between two models.[26] While time-series dependence in returns is sufficiently weak, it

[25] That estimation with different seeds can run independently in parallel limits incremental computing time.

[26] As emphasized by Diebold (2015), the model-free nature of the Diebold-Mariano test means that it should be interpreted as a comparison of forecasts, not as a comparison of "fully articulated econometric models."

is unlikely that the conditions of weak error dependence underlying the Diebold-Mariano test apply to our stock-level analysis due of potentially strong dependence in the cross-section. We adapt Diebold-Mariano to our setting by comparing the cross-sectional average of prediction errors from each model, instead of comparing errors among individual returns. More precisely, to test the forecast performance of method (1) versus (2), we define the test statistic $DM_{12} = \bar{d}_{12}/\widehat{\sigma}_{\bar{d}_{12}}$, where

$$d_{12,t+1} = \frac{1}{n_{3,t+1}} \sum_{i=1}^{n_{3,t+1}} \left(\left(\widehat{e}_{i,t+1}^{(1)} \right)^2 - \left(\widehat{e}_{i,t+1}^{(2)} \right)^2 \right), \tag{20}$$

$\widehat{e}_{i,t+1}^{(1)}$ and $\widehat{e}_{i,t+1}^{(2)}$ denote the prediction error for stock return i at time t using each method, and $n_{3,t+1}$ is the number of stocks in the testing sample (year $t+1$). Then $\bar{d}_{12}$ and $\widehat{\sigma}_{\bar{d}_{12}}$ denote the mean and Newey-West standard error of $d_{12,t}$ over the testing sample. This modified Diebold-Mariano test statistic, which is now based on a single time series $d_{12,t+1}$ of error differences with little autocorrelation, is more likely to satisfy the mild regularity conditions needed for asymptotic normality and in turn provide appropriate p-values for our model comparison tests.

1.9 Variable importance and marginal relationships

Our goal in interpreting machine learning models is modest. We aim to identify covariates that have an important influence on the cross-section of expected returns while simultaneously controlling for the many other predictors in the system.

We discover influential covariates by ranking them according to a notion of variable importance, which we denote as VI_j for the jth input variable. We consider two different notions of importance. The first is the reduction in panel predictive R^2 from setting all values of predictor j to zero, while holding the remaining model estimates fixed (used, e.g., in the context of dimension reduction by Kelly, Pruitt, and Su 2019). The second, proposed in the neural networks literature by Dimopoulos, Bourret, and Lek (1995), is the sum of squared partial derivatives (SSD) of the model to each input variable j, which summarizes the sensitivity of model fits to changes in that variable.[27]

---

[27] In particular, SSD defines the jth variable importance as

$$SSD_j = \sum_{i,t \in \mathcal{T}_1} \left(\frac{\partial g(z;\theta)}{\partial z_j} \Big|_{z=z_{i,t}} \right)^2,$$

where, with a slight abuse of notation, z_j in the denominator of the derivative denotes the j element of the vector of input variables. We measure SSD within the training set, $\mathcal{T}_1$. Note that, because of nondifferentiabilities in tree-based models, the Dimopoulos, Bourret, and Lek (1995) method is not applicable. Therefore, when we conduct this second variable importance analysis, we measure variable importance for random forests and boosted trees using mean decrease in impurity (see, e.g., Friedman 2001).

As part of our analysis, we also trace out the marginal relationship between expected returns and each characteristic. Despite obvious limitations, such a plot is an effective tool for visualizing the first-order impact of covariates in a machine learning model.

2. An Empirical Study of U.S. Equities

2.1 Data and the overarching model

We obtain monthly total individual equity returns from CRSP for all firms listed in the NYSE, AMEX, and NASDAQ. Our sample begins in March 1957 (the start date of the S&P 500) and ends in December 2016, totaling 60 years. The number of stocks in our sample is almost 30,000, with the average number of stocks per month exceeding 6,200.[28] We also obtain the Treasury-bill rate to proxy for the risk-free rate from which we calculate individual excess returns.

In addition, we build a large collection of stock-level predictive characteristics based on the cross-section of stock returns literature. These include 94 characteristics[29] (61 of which are updated annually, 13 are updated quarterly, and 20 are updated monthly). In addition, we include 74 industry dummies corresponding to the first two digits of Standard Industrial Classification (SIC) codes. Table A.6 in the Internet Appendix provides the details of these characteristics.[30]

We also construct eight macroeconomic predictors following the variable definitions detailed in Welch and Goyal (2008), including dividend-price ratio (dp), earnings-price ratio (ep), book-to-market ratio (bm), net equity expansion (ntis), Treasury-bill rate (tbl), term spread (tms), default spread (dfy), and stock variance (svar).[31]

All of the machine learning methods we consider are designed to approximate the overarching empirical model $\mathrm{E}_t(r_{i,t+1}) = g^\star(z_{i,t})$ defined in Equation (2).

---

[28] We include stocks with prices below \$5, share codes beyond 10 and 11, and financial firms. We select the largest possible pool of assets for at least three important reasons. First, these commonly used filters remove certain stocks that are components of the S&P 500 index, and we find it clearly problematic to exclude such important stocks from an asset pricing analysis. Moreover, because we aggregate individual stock return predictions to predict the index, we cannot omit such stocks. Second, our results are less prone to sample selection or data-snooping biases that the literature (see, e.g., Lo and MacKinlay 1990) cautions against. Third, using a larger sample helps avoid overfitting by increasing the ratio of observation count to parameter count. That said, our results are qualitatively identical and quantitively unchanged if we filter out these firms.

[29] We cross-sectionally rank all stock characteristics period-by-period and map these ranks into the [-1,1] interval following Kelly, Pruitt, and Su (2019) and Freyberger, Neuhierl, and Weber (2020).

[30] The ninety-four predictive characteristics are based on those of Green et al. (2017), and we adapt the SAS code available from Jeremiah Green's Web site and extend the sample period back to 1957. Our data construction differs by more closely adhering to variable definitions in original papers. For example, we construct book-equity and operating profitability following Fama and French (2015). Most of these characteristics are released to the public with a delay. To avoid the forward-looking bias, we assume that monthly characteristics are delayed by at most 1 month, quarterly with at least 4 months lag, and annual with at least 6 months lag. Therefore, to predict returns at month $t+1$, we use most recent monthly characteristics at the end of month t, most recent quarterly data by end $t-4$, and most recent annual data by end $t-6$. Another issue is missing characteristics, which we replace with the cross-sectional median at each month for each stock, respectively.

[31] The monthly data are available from Amit Goyal's Web site.

Throughout our analysis, we define the baseline set of stock-level covariates $z_{i,t}$ as

$$z_{i,t} = x_t \otimes c_{i,t}, \tag{21}$$

where $c_{i,t}$ is a $P_c \times 1$ matrix of characteristics for each stock i, and x_t is a $P_x \times 1$ vector of macroeconomic predictors (and are thus common to all stocks, including a constant). Thus, $z_{i,t}$ is a $P \times 1$ vector of features for predicting individual stock returns (with $P = P_c P_x$) and includes interactions between stock-level characteristics and macroeconomic state variables. The total number of covariates is $94 \times (8+1)+74 = 920$.

The overarching model specified by (2) and (21) nests many models proposed in the literature (Rosenberg 1974; Harvey and Ferson 1999, among others). The motivating example for this model structure is the standard beta pricing representation of the asset pricing conditional Euler equation,

$$\mathrm{E}_t(r_{i,t+1}) = \beta_{i,t}' \lambda_t. \tag{22}$$

The structure of our feature set in (21) allows for purely stock-level information to enter expected returns via $c_{i,t}$ in analogy with the risk exposure function $\beta_{i,t}$, and also allows aggregate economic conditions to enter in analogy with the dynamic risk premium λ_t. In particular, if $\beta_{i,t} = \theta_1 c_{i,t}$, and $\lambda_t = \theta_2 x_t$, for some constant parameter matrices θ_1 ($K \times P_c$) and θ_2 ($K \times P_x$), then the beta pricing model in (22) becomes

$$g^\star(z_{i,t}) = \mathrm{E}_t(r_{i,t+1}) = \beta_{i,t}' \lambda_t = c_{i,t}' \theta_1' \theta_2 x_t = \left(x_t \otimes c_{i,t}\right)' \mathrm{vec}(\theta_1' \theta_2) =: z_{i,t}' \theta, \tag{23}$$

where $\theta = \mathrm{vec}(\theta_1' \theta_2)$. The overarching model is more general than this example because $g^\star(\cdot)$ is not restricted to be a linear function. Considering nonlinear $g^\star(\cdot)$ formulations, for example, via generalized linear models or neural networks, essentially expands the feature set to include a variety of functional transformations of the baseline $z_{i,t}$ predictor set.

We divide the 60 years of data into 18 years of training sample (1957–1974), 12 years of validation sample (1975–1986), and the remaining 30 years (1987–2016) for out-of-sample testing. Because machine learning algorithms are computationally intensive, we avoid recursively refitting models each month. Instead, we refit once every year as most of our signals are updated once per year. Each time we refit, we increase the training sample by 1 year. We maintain the same size of the validation sample, but roll it forward to include the most recent 12 months.[32]

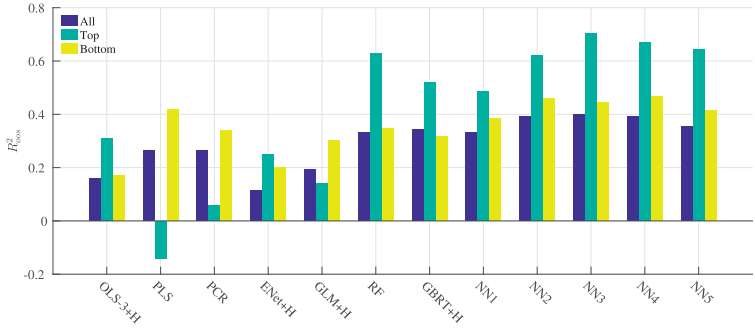
2.2 The cross-section of individual stocks

Table 1 presents the comparison of machine learning techniques in terms of their out-of-sample predictive R^2. We compare thirteen models in total, including

[32] We do not use cross-validation to maintain the temporal ordering of the data.

Table 1
Monthly out-of-sample stock-level prediction performance (percentage R^2_{oos})

	OLS +H	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
All	−3.46	0.16	0.27	0.26	0.11	0.19	0.33	0.34	0.33	0.39	0.40	0.39	0.36
Top 1,000	−11.28	0.31	−0.14	0.06	0.25	0.14	0.63	0.52	0.49	0.62	0.70	0.67	0.64
Bottom 1,000	−1.30	0.17	0.42	0.34	0.20	0.30	0.35	0.32	0.38	0.46	0.45	0.47	0.42



In this table, we report monthly R^2_{oos} for the entire panel of stocks using OLS with all variables (OLS), OLS using only size, book-to-market, and momentum (OLS-3), PLS, PCR, elastic net (ENet), generalize linear model (GLM), random forest (RF), gradient boosted regression trees (GBRT), and neural networks with 1 to 5 layers (NN1–NN5). "+H" indicates the use of Huber loss instead of the l_2 loss. We also report these R^2_{oos} within subsamples that include only the top-1,000 stocks or bottom-1,000 stocks by market value. The lower panel provides a visual comparison of the R^2_{oos} statistics in the table (omitting OLS because of its large negative values).

OLS with all covariates, OLS-3 (which preselects size, book-to-market, and momentum as the only covariates), PLS, PCR, elastic net (ENet), generalized linear model with group lasso (GLM), random forest (RF), gradient boosted regression trees (GBRT), and neural network architectures with one to five layers (NN1,...,NN5). For OLS, ENet, GLM, and GBRT, we present their robust versions using Huber loss, which perform better than the version without.

The first row of Table 1 reports R^2_{oos} for the entire pooled sample. The OLS model using all 920 features produces an R^2_{oos} of -3.46%, indicating it is handily dominated by applying a naive forecast of zero to all stocks in all months. This may be unsurprising as the lack of regularization leaves OLS highly susceptible to in-sample overfit. However, restricting OLS to a sparse parameterization, either by forcing the model to include only three covariates (size, value, and momentum), or by penalizing the specification with the elastic net—generates a substantial improvement over the full OLS model (R^2_{oos} of 0.16% and 0.11% respectively). Figure 3 summarizes the complexity of each model at each reestimation date. The upper left panel shows the number of features to which elastic net assigns a nonzero loading. In the first 10 years of the test sample, the model typically chooses fewer than five features. After 2,000, the number of selected features rises and hovers between 20 and 40.

Regularizing the linear model via dimension reduction improves predictions even further. By forming a few linear combinations of predictors, PLS and

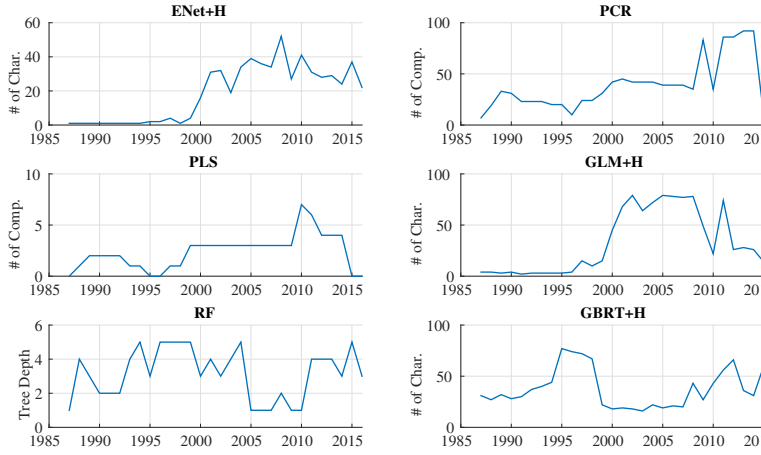


Figure 3
Time-varying model complexity
This figure demonstrates the model's complexity for elastic net (ENet), PCR, PLS, generalized linear model with group lasso (GLM), random forest (RF), and gradient boosted regression trees (GBRT) in each training sample of our 30-year recursive out-of-sample analysis. For ENet and GLM, we report the number of features selected to have nonzero coefficients; for PCR and PLS, we report the number of selected components; for RF, we report the average tree depth; and, for GBRT, we report the number of distinct characteristics entering into the trees.

especially PCR, raise the out-of-sample R^2 to 0.27% and 0.26%, respectively. Figure 3 shows that PCR typically uses 20 to 40 components in its forecasts. PLS, on the other hand, fails to find a single reliable component for much of the early sample, but eventually settles on three to six components. The improvement of dimension reduction over variable selection via elastic net suggests that characteristics are partially redundant and fundamentally noisy signals. Combining them into low-dimension components averages out noise to better reveal their correlated signals.

The generalized linear model with group lasso penalty fails to improve on the performance of purely linear methods (R^2_{oos} of 0.19%). The fact that this method uses spline functions of individual features, but includes no interaction among features, suggests that univariate expansions provide little incremental information beyond the linear model. Though it tends to select more features than elastic net, those additional features do not translate into incremental performance.

Boosted trees and random forests are competitive with PCR, producing fits of 0.34% and 0.33%, respectively. Random forests generally estimate shallow trees, with one to five layers on average. To quantify the complexity of GBRT, we report the number of features used in the boosted tree ensemble at each reestimation point. In the beginning of the sample GBRT uses around 30 features to partition outcomes, with this number increasing to 50 later in the sample.

Neural networks are the best performing nonlinear method, and the best predictor overall. The R^2_{oos} is 0.33% for NN1 and peaks at 0.40% for NN3. These results point to the value of incorporating complex predictor interactions, which are embedded in tree and neural network models but that are missed by other techniques. The results also show that in the monthly return setting, the benefits of "deep" learning are limited, as four- and five-layer models fail to improve over NN3.[33]

The second and third rows of Table 1 break out predictability for large stocks (the top-1,000 stocks by market equity each month) and small stocks (the bottom-1,000 stocks each month). This is based on the full estimated model (using all stocks), but focuses on fits among the two subsamples. The baseline patterns that OLS fares poorly, regularized linear models are an improvement, and nonlinear models dominate carries over into subsamples. Tree methods and neural networks are especially successful among large stocks, with R^2_{oos} ranging from 0.52% to 0.70%. This dichotomy provides reassurance that machine learning is not merely picking up small scale inefficiencies driven by illiquidity.[34]

Table 2 conducts our analysis at the annual horizon. The comparative performance across different methods is similar to the monthly results shown in Table 1, but the annual R^2_{oos} is nearly an order of magnitude larger. Their success in forecasting annual returns likewise illustrates that machine learning models are able to isolate risk premiums that persist over business cycle frequencies and are not merely capturing short-lived inefficiencies.

Whereas Table 1 offers a quantitative comparison of models' predictive performance, Table 3 assesses the statistical significance of differences among models at the monthly frequency. It reports Diebold-Mariano test statistics for pairwise comparisons of a column model versus a row model. Diebold-Mariano statistics are distributed $\mathcal{N}(0,1)$ under the null of no difference between models, thus the test statistic magnitudes map to p-values in the same way as regression t-statistics. Our sign convention is that a positive statistic indicates the column model outperforms the row model. Bold numbers denote significance at the 5% level for each individual test.

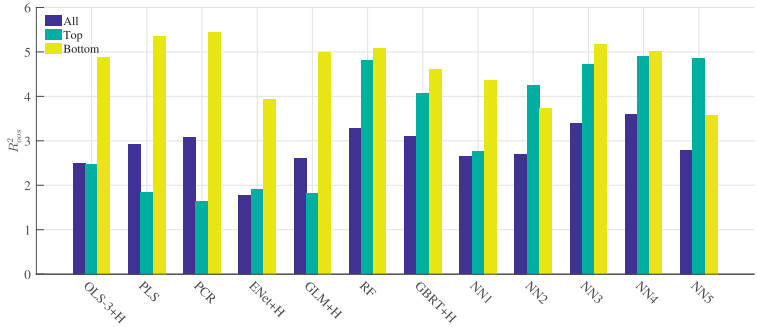
The first conclusion from Table 3 is that constrained linear models—including restricting OLS to only use three predictors, reducing dimension via PLS or PCA, and penalizing via elastic net—produce statistically significant improvements over the unconstrained OLS model. Second, we see little difference in the performance of penalized linear methods and dimension reduction methods. Third, we find that tree-based methods uniformly improve

[33] Because we hold the five neural networks architectures fixed and simply compare across them, we do not describe their estimated complexity in Figure 3.

[34] As an aside, it is useful to know that there is a roughly 3% inflation in out-of-sample R^2s if performance is benchmarked against historical averages. For OLS-3, the R^2 relative to the historical mean forecast is 3.74% per month! Evidently, the historical mean is such a noisy forecaster that it is easily beaten by a fixed excess return forecasts of zero.

Table 2
Annual out-of-sample stock-level prediction performance (percentage R^2_{oos})

	OLS +H	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
All	−34.86	2.50	2.93	3.08	1.78	2.60	3.28	3.09	2.64	2.70	3.40	3.60	2.79
Top	−54.86	2.48	1.84	1.64	1.90	1.82	4.80	4.07	2.77	4.24	4.73	4.91	4.86
Bottom	−19.22	4.88	5.36	5.44	3.94	5.00	5.08	4.61	4.37	3.72	5.17	5.01	3.58



Annual return forecasting R^2_{oos} (see the legend to Table 1).

Table 3
Comparison of monthly out-of-sample prediction using Diebold-Mariano tests

	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
OLS+H	**3.26***	**3.29***	**3.35***	**3.29***	**3.28***	**3.29***	**3.26***	**3.34***	**3.40***	**3.38***	**3.37***	**3.38***
OLS-3+H		1.42	**1.87**	−0.27	0.62	**1.64**	1.28	1.25	**2.13**	**2.13**	**2.36**	**2.11**
PLS			−0.19	−1.18	−1.47	0.87	0.67	0.63	1.32	1.37	1.66	1.08
PCR				−1.10	−1.37	0.85	0.75	0.58	1.17	1.19	1.34	1.00
ENet+H					0.64	**1.90**	1.40	**1.73**	**1.97**	**2.07**	**1.98**	**1.85**
GLM+H						**1.76**	1.22	1.29	**2.28**	**2.17**	**2.68***	**2.37**
RF							0.07	−0.03	0.31	0.37	0.34	0.00
GBRT+H								−0.06	0.16	0.21	0.17	−0.04
NN1									0.56	0.59	0.45	0.04
NN2										0.32	−0.03	−0.88
NN3											−0.32	−0.92
NN4												−1.04

This table reports pairwise Diebold-Mariano test statistics comparing the out-of-sample stock-level prediction performance among thirteen models. Positive numbers indicate the column model outperforms the row model. Bold font indicates the difference is significant at 5% level or better for individual tests, and an asterisk indicates significance at the 5% level for 12-way comparisons via our conservative Bonferroni adjustment.

over linear models, but the improvements are at best marginally significant. Neural networks are the only models that produce large and significant statistical improvements over linear and generalized linear models. They also improve over tree models, but the difference is not statistically significant.

Table 3 makes multiple comparisons. We highlight how inference changes under a conservative Bonferroni multiple comparisons correction that divides

the significance level by the number of comparisons.[35] For a significance level of 5% amid 12 model comparisons, the adjusted one-sided critical value in our setting is 2.64. In the table, tests that exceed this conservative threshold are accompanied by an asterisk. The main difference with a Bonferroni adjustment is that neural networks become only marginally significant over penalized linear models.

2.3 Which covariates matter?

We now investigate the relative importance of individual covariates for the performance of each model using the importance measures described in Section 1.9. To begin, for each method, we calculate the reduction in R^2 from setting all values of a given predictor to zero within each training sample, and average these into a single importance measure for each predictor. Figure 4 reports the resultant importance of the top-20 stock-level characteristics for each method. Variable importance within the model is normalized to sum to one, allowing for the interpretation of relative importance for that particular model.

Figure 5 reports overall rankings of characteristics for all models. We rank the importance of each characteristic for each method, then sum their ranks. Characteristics are ordered so that the highest total ranks are on top and the lowest ranking characteristics are at the bottom. The color gradient within each column shows the model-specific ranking of characteristics from least to most important (lightest to darkest).[36]

Figures 4 and 5 demonstrate that models are generally in close agreement regarding the most influential stock-level predictors, which can be grouped into four categories. The first are based on recent price trends, including 5 of the top-7 variables in Figure 5: short-term reversal (mom1m), stock momentum (mom12m), momentum change (chmom), industry momentum (indmom), recent maximum return (maxret), and long-term reversal (mom36m). Next are liquidity variables, including turnover and turnover volatility (turn, SD_turn), log market equity (mvel1), dollar volume (dolvol), Amihud illiquidity (ill), number of zero trading days (zerotrade), and bid-ask spread (baspread). Risk measures constitute the third influential group, including total and idiosyncratic return volatility (retvol, idiovol), market beta (beta), and beta squared (betasq). The last group includes valuation ratios and fundamental signals, such as earnings-to-price (ep), sales-to-price (sp), asset growth (agr), and number of

[35] Multiple comparisons are a concern when the researcher conducts many hypotheses tests and draws conclusions based on only those that are significant. Doing so distorts the size of tests through a selection bias. Instead, we report t-statistics for every comparison we consider, yet we report adjusted inference to err on the side of caution. We also note that false discoveries in multiple comparisons should be randomly distributed. The statistically significant t-statistics in our analyses do not appear random, but instead follow a pattern in which nonlinear models outperform linear ones.

[36] Figure 5 is based on the average rank of over thirty recursing training samples. Figure A.1 in the Internet Appendix presents the ranks for each of the recursing samples, respectively. The rank of important characteristics (top-third of the covariates) is remarkably stable over time. This is true for all models, though we show results for one representative model (NN3) in the interest of space.

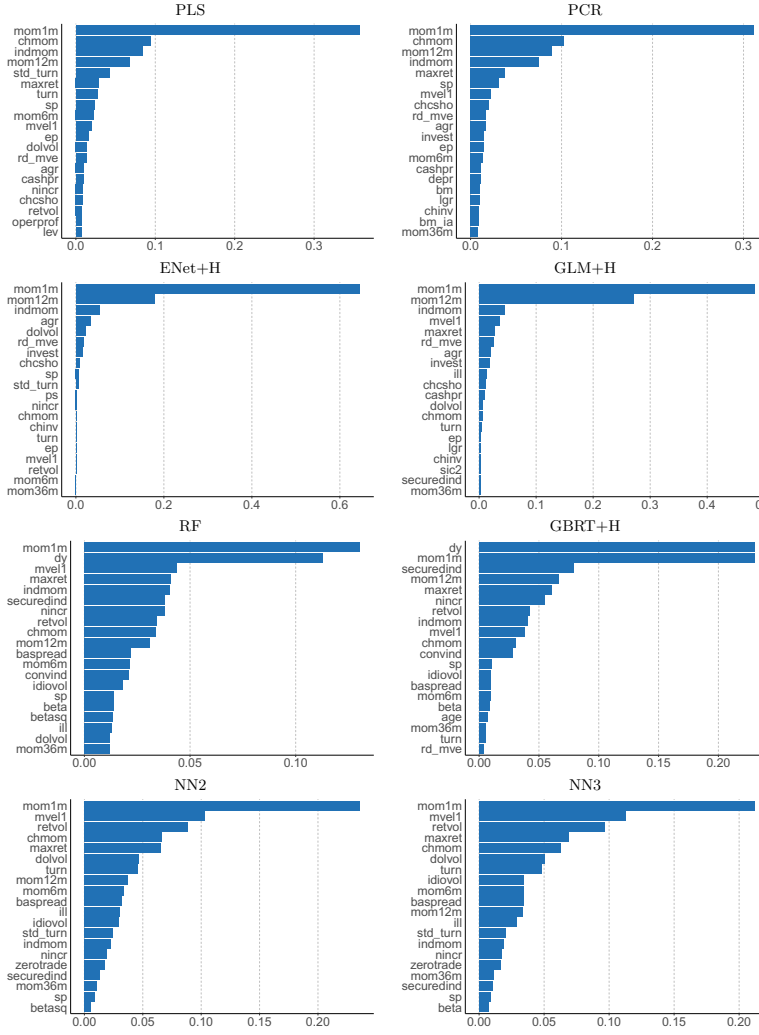


Figure 4
Variable importance by model
Variable importance for the top-20 most influential variables in each model. Variable importance is an average over all training samples. Variable importance within each model is normalized to sum to one.

recent earnings increases (nincr). Figure 4 shows that characteristic importance magnitudes for penalized linear models and dimension reduction models are highly skewed toward momentum and reversal. Trees and neural networks are more democratic, drawing predictive information from a broader set of characteristics.

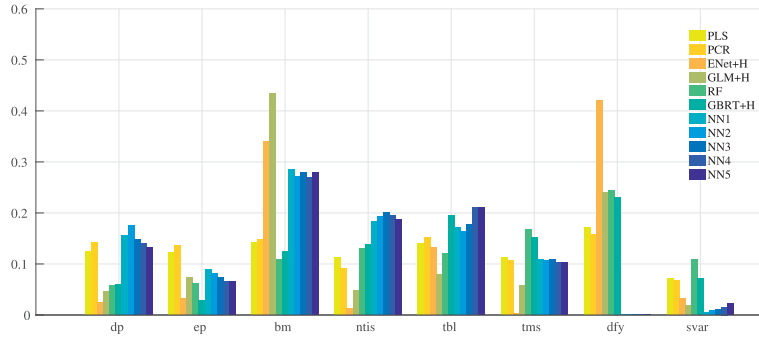
We find that our second measure of variable importance, SSD from Dimopoulos, Bourret, and Lek (1995), produces very similar results to the

Figure 5
Characteristic importance
Rankings of ninety-four stock-level characteristics and the industry dummy (sic2) in terms of overall model contribution. Characteristics are ordered based on the sum of their ranks over all models, with the most influential characteristics on the top and the least influential on the bottom. Columns correspond to the individual models, and the color gradients within each column indicate the most influential (dark blue) to the least influential (white) variables.

simpler R^2 measure. Within each model, we calculate the Pearson correlation between relative importance from SSD and the R^2 measure. These correlations range from 84.0% on the low end (NN1) to 97.7% on the high end (random forest). That is, the two methods provide a highly consistent summary of which variables are most influential for forecast accuracy. Figure A.2 in the Internet Appendix shows the full set of SSD results.

Table 4
Variable importance for macroeconomic predictors

	PLS	PCR	ENet+H	GLM+H	RF	GBRT+H	NN1	NN2	NN3	NN4	NN5
dp	12.52	14.12	2.49	4.54	5.80	6.05	15.57	17.58	14.84	13.95	13.15
ep	12.25	13.52	3.27	7.37	6.27	2.85	8.86	8.09	7.34	6.54	6.47
bm	14.21	14.83	33.95	43.46	10.94	12.49	28.57	27.18	27.92	26.95	27.90
ntis	11.25	9.10	1.30	4.89	13.02	13.79	18.37	19.26	20.15	19.59	18.68
tbl	14.02	15.29	13.29	7.90	11.98	19.49	17.18	16.40	17.76	20.99	21.06
tms	11.35	10.66	0.31	5.87	16.81	15.27	10.79	10.59	10.91	10.38	10.33
dfy	17.17	15.68	42.13	24.10	24.37	22.93	0.09	0.06	0.06	0.04	0.12
svar	7.22	6.80	3.26	1.87	10.82	7.13	0.57	0.85	1.02	1.57	2.29



Variable importance for eight macroeconomic variables in each model. Variable importance is an average over all training samples. Variable importance within each model is normalized to sum to one. The lower panel provides a complementary visual comparison of macroeconomic variable importance.

For robustness, we rerun our analysis with an augmented set of characteristics that include five placebo "characteristics." They are simulated according to the data-generating process (A.1) in Internet Appendix A. The parameters are calibrated to have similar behavior as our characteristics data set but are independent of future returns by construction. Figure A.3 in Internet Appendix F presents the variable importance plot (based on R^2) with five noise characteristics highlighted. The table confirms that the most influential characteristics that we identify in our main analysis are unaffected by the presence of irrelevant characteristics. Noise variables appear among the least informative characteristics, along with sin stocks, dividend initiation/omission, cashflow volatility, and other accounting variables.

Table 4 shows the R^2-based importance measure for each macroeconomic predictor variable (again normalized to sum to one within a given model). All models agree that the aggregate book-to-market ratio is a critical predictor, whereas market volatility has little role in any model. PLS and PCR place similar weights on all other predictors, potentially because these variables are highly correlated. Linear and generalized linear models strongly favor bond market variables, including the default spread and Treasury rate. Nonlinear methods (trees and neural networks) place great emphasis on exactly those predictors ignored by linear methods, such as term spreads and issuance activity.

Many accounting characteristics are not available at the monthly frequency, which might explain their low importance in Figure 5. To investigate this,

Figure A.6 in the Internet Appendix presents the rank of variables based on the annual return forecasts. Price trend variables become less important compared to the liquidity and risk measures, although they are still quite influential. The characteristics that were ranked in the bottom half of predictors at the monthly horizon remain largely unimportant at the annual horizon. The exception is industry (sic2), which shows substantial predictive power at the annual frequency.

2.3.1 Marginal association between characteristics and expected returns.. Figure 6 traces out the model-implied marginal impact of individual characteristics on expected excess returns. Our data transformation normalizes characteristics to the (-1,1) interval, and holds all other variables fixed at their median value of zero. We choose four illustrative characteristics for the figure, including size (mvel1), momentum (mom12m), stock volatility (retvol), and accruals (acc).

First, Figure 6 illustrates that machine learning methods identify patterns similar to some well-known empirical phenomena. For example, expected stock returns are decreasing in size, increasing in past 1-year return, and decreasing in stock volatility. And, interestingly, all methods agree on a nearly exact zero relationship between accruals and future returns. Second, the (penalized) linear model finds no predictive association between returns and either size or volatility, while trees and neural networks find large sensitivity of expected returns to both of these variables. For example, a firm that drops from median size to the 20th percentile of the size distribution experiences an increase in its annualized expected return of roughly 2.4% ($0.002\times12\times100$), and a firm whose volatility rises from median to 80th percentile experiences a decrease of around 3.0% per year, according to NN3, and these methods detect nonlinear predictive associations. The inability of linear models to capture nonlinearities can lead them to prefer a zero association, and this can in part explain the divergence in the performance of linear and nonlinear methods.

2.3.2 Interaction effects. The favorable performance of trees and neural networks indicates a benefit to allowing for potentially complex interactions among predictors. Machine learning models are often referred to as "black boxes," a term that is in some sense a misnomer, as the models are readily inspectable. They are, however, complex, and this is the source of both their power and their opacity. Any exploration of interaction effect is vexed by vast possibilities for identity and functional forms for interacting predictors. In this section, we present a handful of interaction results to help illustrate the inner workings of one black box method, the NN3 model.

As a first example, we examine a set of pairwise interaction effects in NN3. Figure 7 reports how expected returns vary as we simultaneously vary values of a pair of characteristics over their support [-1,1], while holding all other variables fixed at their median value of zero. We show interactions of

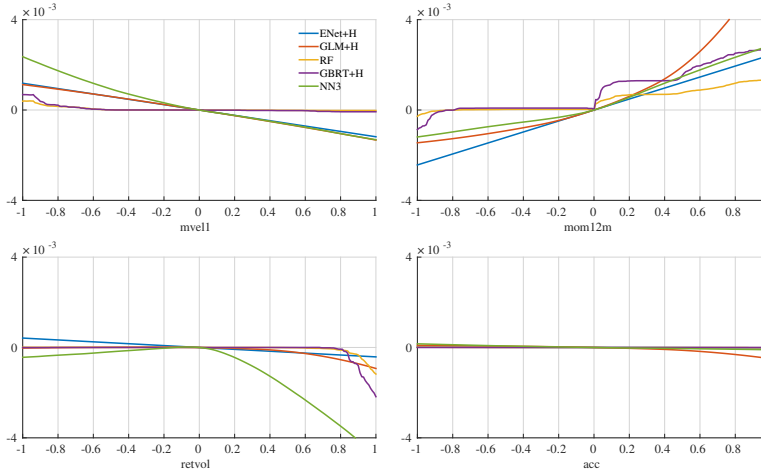


Figure 6
Marginal association between expected returns and characteristics
The panels should the sensitivity of expected monthly percentage returns (vertical axis) to the individual characteristics (holding all other covariates fixed at their median values).

stock size (mvel1) with four other predictors: short-term reversal (mom1m), momentum (mom12m), and total and idiosyncratic volatility (retvol and idiovol, respectively).

The upper-left figure shows that the short-term reversal effect is strongest and is essentially linear among small stocks (blue line). Among large stocks (green line), reversal is concave, occurring primarily when the prior month return is positive. The upper-right figure shows the momentum effect, which is most pronounced among large stocks for the NN3 model.[37] Likewise, on the lower-left side, we see that the low volatility anomaly is also strongest among large stocks. For small stocks, the volatility effect is hump shaped. Finally, the lower-right side shows that NN3 estimates no interaction effect between size and accruals—the size lines are simply vertical shifts of the univariate accruals curve.

Figure 8 illustrates interactions between stock-level characteristics and macroeconomic indicator variables. It shows, for example, that the size effect is more pronounced when aggregate valuations are low (bm is high) and when equity issuance (ntis) is low, while the low volatility anomaly is especially strong in high valuation and high issuance environments. Figure A.4 in the Internet Appendix shows the 100 most important interactions of stock characteristics with macroeconomic indicators for each machine learning model. The most influential features come from interacting a stock's recent

[37] Conclusions from our model can diverge from the results in the literature, because we jointly model hundreds of predictor variables, which can lead to new conclusions regarding marginal effects, interaction effects, and so on.

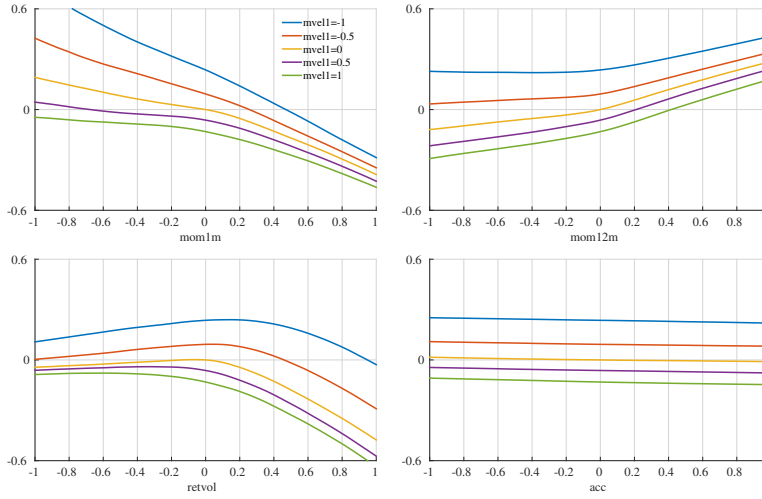


Figure 7
Expected returns and characteristic interactions (NN3)
The panels should the sensitivity of the expected monthly percentage returns (vertical axis) to the interactions effects for mvel1 with mom1m, mom12m, retvol, and acc in model NN3 (holding all other covariates fixed at their median values).

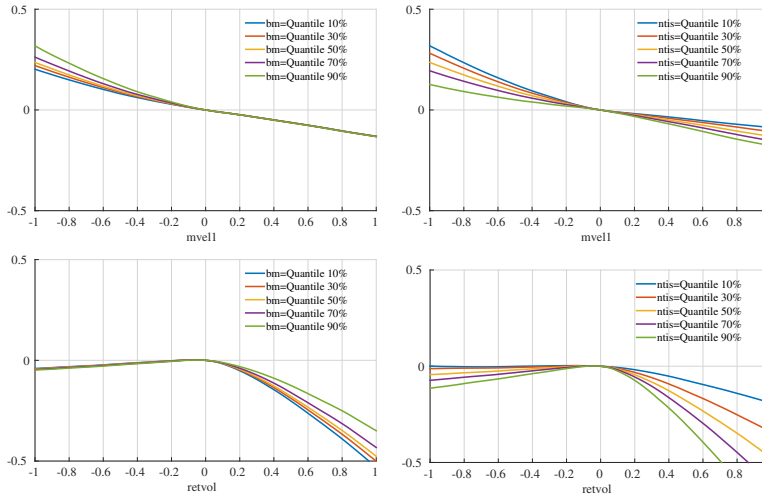


Figure 8
Expected returns and characteristic/macroeconomic variable interactions (NN3)
The panels show the sensitivity of expected monthly percentage returns (vertical axis) to interactions effects for mvel1 and retvol with bm and ntis in model NN3 (holding all other covariates fixed at their median values).

price trends (e.g., momentum, short-term reversal, or industry momentum) with aggregate asset price levels (e.g., valuation ratios of the aggregate stock market or Treasury-bill rates). Furthermore, the dominant macroeconomic interactions are stable over time, as illustrated in Figure A.5 in the Internet Appendix.

2.4 Portfolio forecasts

So far, we have analyzed predictability of individual stock returns. Next, we compare forecasting performance of machine learning methods for aggregate portfolio returns. Analyzing forecasts at the portfolio-level comes with a number of benefits.

First, because all of our models are optimized for stock-level forecasts, portfolio forecasts provide an additional indirect evaluation of the model and its robustness. Second, aggregate portfolios tend to be of broader economic interest because they represent the risky-asset savings vehicles most commonly held by investors (via mutual funds, ETFs, and hedge funds). We study value-weight portfolios to assess the extent to which a model's predictive performance thrives in the most valuable (and most economically important) assets in the economy. Third, the distribution of portfolio returns is sensitive to dependence among stock returns, with the implication that a good stock-level prediction model is not guaranteed to produce accurate portfolio-level forecasts. Bottom-up portfolio forecasts allow us to evaluate a model's ability to transport its asset predictions, which occur at the finest asset level, into broader investment contexts. Last, but not least, the portfolio results are one step further "out of sample" in that the optimization routine does not directly account for the predictive performance of the portfolios.

Our assessment of forecast performance up to this point has been entirely statistical, relying on comparisons of predictive R^2. The final advantage of analyzing predictability at the portfolio level is that we can assess the *economic* contribution of each method via its contribution to risk-adjusted portfolio return performance.

2.4.1 Prespecified portfolios. We build bottom-up forecasts by aggregating individual stock return predictions into portfolios. Given the weight of stock i in portfolio p (denoted $w_{i,t}^p$) and given a model-based out-of-sample forecast for stock i (denoted $\widehat{r}_{i,t+1}$), we construct the portfolio return forecast as

$$\widehat{r}_{t+1}^p = \sum_{i=1}^{n} w_{i,t}^p \times \widehat{r}_{i,t+1}.$$

This bottom-up approach works for any target portfolio whose weights are known a priori.

We form bottom-up forecasts for 30 of the most well-known portfolios in the empirical finance literature, including the S&P 500, the Fama-French size, value, profitability, investment, and momentum factor portfolios (SMB,

HML, RMW, CMA, and UMD, respectively), and subcomponents of these Fama-French portfolios, including six size and value portfolios, six size and investment portfolios, six size and profitability portfolios, and six size and momentum portfolios. The subcomponent portfolios are long-only and therefore substantially overlap with the market index, whereas SMB, HML, RMW, CMA, and UMD are zero-net-investment long-short portfolios that are in large part purged of market exposure. In all cases, we create the portfolios ourselves using CRSP market equity value weights. Our portfolio construction differs slightly from the actual S&P 500 index and the characteristic-based Fama-French portfolios (Fama and French 1993, 2015), but has the virtue that we can exactly track the ex ante portfolio weights of each.[38]

Table 5 reports the monthly out-of-sample R^2 over our 30-year testing sample. Regularized linear methods fail to outperform naive constant forecasts of the S&P 500. In contrast, all nonlinear models have substantial positive predictive performance. The 1-month out-of-sample R^2 is 0.71% for the generalized linear model and reaches as high as 1.80% for the three-layer neural network. As a benchmark for comparison, nearly all of the macroeconomic return predictor variables in the survey of Welch and Goyal (2008) fail to produce a positive out-of-sample forecast R^2. Kelly and Pruitt (2013) find that PLS delivers an out-of-sample forecasting R^2 around 1% per month for the aggregate market index, though their forecasts directly target the market return as opposed to being bottom-up forecasts. And the most well-studied portfolio predictors, such as the aggregate price-dividend ratio, typically produce an *in-sample* predictive R^2 of around 1% per month (e.g., Cochrane 2007), smaller than what we find *out of sample*.

The patterns in S&P 500 forecasting performance across models carry over to long-only characteristic-sorted portfolios (rows 2–25) and long-short factor portfolios (SMB, HML, RMW, CMA, and UMD). Nonlinear methods excel. NN3–NN5 produce a positive R^2_{oos} for *every* portfolio analyzed, with NN3 dominating. NN1 and NN2 also produce a positive R^2_{oos} for all but UMD (which is by and large the most difficult strategy to forecast). The generalized linear model R^2_{oos} is positive for 22 of 30 portfolios. Linear methods, on the other hand, are on balance unreliable for bottom-up portfolio return forecasting, though their performance tends to be better for "big" portfolios compared to "small." For select long-short factor portfolios (e.g., SMB), constrained linear methods, such as PLS, perform comparatively well (consistent with the findings of Kelly and Pruitt 2013). In short, machine learning methods and nonlinear methods, in particular, produce unusually powerful out-of-sample portfolio predictions.

Next, we assess the economic magnitudes of portfolio predictability. Campbell and Thompson (2008) show that small improvements in R^2 can map

[38] Our replication of S&P 500 returns has a correlation with the actual index of more than 0.99. For the other portfolios (SMB, HML, RMW, CMA, UMD), the return correlations between our replication and the version from Kenneth French's Web site are 0.99, 0.97, 0.95, 0.99, and 0.96, respectively.

Table 5
Monthly portfolio-level out-of-sample predictive R^2

	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
A. Common factor portfolios												
S&P 500	−0.22	−0.86	−1.55	0.75	0.71	1.37	1.40	1.08	1.13	1.80	1.63	1.17
SMB	0.81	2.09	0.39	1.72	2.36	0.57	0.35	1.40	1.16	1.31	1.20	1.27
HML	0.66	0.50	1.21	0.46	0.84	0.98	0.21	1.22	1.31	1.06	1.25	1.24
RMW	−2.35	1.19	0.41	−1.07	−0.06	−0.54	−0.92	0.68	0.47	0.84	0.53	0.54
CMA	0.80	−0.44	0.03	−1.07	1.24	−0.11	−1.04	1.88	1.60	1.06	1.84	1.31
UMD	−0.90	−1.09	−0.47	0.47	−0.37	1.37	−0.25	−0.56	−0.26	0.19	0.27	0.35
B. Subcomponents of factor portfolios												
Big value	0.10	0.00	−0.33	0.25	0.59	1.31	1.06	0.85	0.87	1.46	1.21	0.99
Big growth	−0.33	−1.26	−1.62	0.70	0.51	1.32	1.19	1.00	1.10	1.50	1.24	1.11
Big neutral	−0.17	−1.09	−1.51	0.80	0.36	1.31	1.28	1.43	1.24	1.70	1.81	1.40
Small value	0.30	1.66	1.05	0.64	0.85	1.24	0.52	1.59	1.37	1.54	1.40	1.30
Small growth	−0.16	0.14	−0.18	−0.33	−0.12	0.71	1.24	0.05	0.42	0.48	0.41	0.50
Small neutral	−0.27	0.60	0.19	0.21	0.28	0.88	0.36	0.58	0.62	0.70	0.58	0.68
Big conservative	−0.57	−0.10	−1.06	1.02	0.46	1.11	0.55	1.15	1.13	1.59	1.37	1.07
Big aggressive	0.20	−0.80	−1.15	0.30	0.67	1.75	2.00	1.33	1.51	1.78	1.55	1.42
Big neutral	−0.29	−1.75	−1.96	0.83	0.48	1.13	0.77	0.85	0.85	1.51	1.45	1.16
Small conservative	−0.05	1.17	0.71	−0.02	0.34	0.96	0.56	0.82	0.87	0.96	0.90	0.83
Small aggressive	−0.10	0.51	0.01	−0.09	0.14	1.00	1.46	0.34	0.64	0.75	0.62	0.71
Small neutral	−0.30	0.45	0.12	0.42	0.35	0.76	−0.01	0.70	0.69	0.83	0.66	0.72
Big robust	−1.02	−1.08	−2.06	0.55	0.35	1.10	0.33	0.74	0.79	1.28	1.03	0.74
Big weak	−0.12	1.42	1.07	0.89	1.10	1.33	1.77	1.79	1.79	2.05	1.66	1.60
Big neutral	0.86	−1.22	−1.26	0.41	0.13	1.10	0.91	0.84	0.94	1.19	1.15	0.99
Small robust	−0.71	0.35	−0.38	−0.04	−0.42	0.70	0.19	0.24	0.50	0.63	0.53	0.55
Small weak	0.05	1.06	0.59	−0.13	0.44	1.05	1.42	0.71	0.92	0.99	0.90	0.89
Small neutral	−0.51	0.07	−0.47	−0.33	−0.32	0.60	−0.08	0.10	0.25	0.38	0.32	0.41
Big up	0.20	−0.25	−1.24	0.66	1.17	1.18	0.90	0.80	0.76	1.13	1.12	0.93
Big down	−1.54	−1.63	−1.55	0.44	−0.33	1.14	0.71	0.36	0.70	1.07	0.90	0.84
Big medium	−0.04	−1.51	−1.94	0.81	−0.08	1.57	1.80	1.29	1.32	1.71	1.55	1.23
Small up	0.07	0.78	0.56	−0.07	0.25	0.62	−0.03	0.06	0.07	0.21	0.19	0.25
Small down	−0.21	0.15	−0.20	0.15	−0.01	1.51	1.38	0.74	0.82	1.02	0.91	0.96
Small medium	0.07	0.82	0.20	0.59	0.37	1.22	1.06	1.09	1.09	1.18	1.00	1.03

In this table, we report the out-of-sample predictive R^2s for thirty portfolios using OLS with size, book-to-market, momentum, OLS-3, PLS, PCR, elastic net (ENet), generalized linear model with group lasso (GLM), random forest (RF), gradient boosted regression trees (GBRT), and five architectures of neural networks (NN1,...,NN5). "+H" indicates the use of Huber loss instead of the l_2 loss. The six portfolios in panel A are the S&P 500 index and the Fama-French SMB, HML, CMA, RMW, and UMD factors. The twenty-four portfolios in panel B are 3×2 size double-sorted portfolios used in the construction of the Fama-French value, investment, profitability, and momentum factors.

into large utility gains for a mean-variance investor. They show that the Sharpe ratio (SR^*) earned by an active investor exploiting predictive information (summarized as a predictive R^2) improves over the Sharpe ratio (SR) earned by a buy-and-hold investor according to

$$SR^* = \sqrt{\frac{SR^2 + R^2}{1 - R^2}}.$$

We use this formula to translate the predictive R^2_{oos} of Table 5 (along with the full-sample Sharpe ratio of each portfolio) into an improvement in annualized

Sharpe ratio, $SR^* - SR$, for an investor exploiting machine learning predictions for portfolio timing. Table A.7 in the Internet Appendix presents the results. For example, the buy-and-hold Sharpe ratio of the S&P 500, which is 0.51 in our 30-year out-of-sample period, can be improved to 0.71 by a market-timer exploiting forecasts from the NN3 model. For characteristic-based portfolios, nonlinear machine learning methods help improve Sharpe ratios by anywhere from a few percentage points to over 24 percentage points.

Campbell and Thompson (2008) also propose evaluating the economic magnitude of portfolio predictability with a market timing trading strategy. We follow their out-of-sample trading strategy exactly, scaling up/down positions each month as expected returns rise/fall, while imposing a maximum leverage constraint of 50% and excluding short sales for long-only portfolios (we do not impose these constraints for long-short factor portfolios). Table 6 reports the annualized Sharpe ratio gains (relative to a buy-and-hold strategy) for timing strategies based on machine learning forecasts. Consistent with our other results, the strongest and most consistent trading strategies are those based on nonlinear models, with neural networks the best overall. In the case of NN3, the Sharpe ratio from timing the S&P 500 index 0.77, or 26 percentage points higher than a buy-and-hold position. Long-short factor portfolios are more difficult to time than the S&P 500 and the other 24 long-only portfolios, and all methods fail to outperform the static UMD strategy.

As a robustness test, we also analyze bottom-up predictions for annual rather than monthly returns. The comparative patterns in predictive performance across methods is the same in annual and monthly data. Table A.8 in the Internet Appendix demonstrates the superiority of nonlinear methods and in particular neural networks. NN3 continues to dominate for the market portfolio, achieving an annual R^2_{oos} of 15.7%.

2.4.2 Machine learning portfolios. Next, rather than assessing forecast performance among prespecified portfolios, we design a new set of portfolios to directly exploit machine learning forecasts. At the end of each month, we calculate 1-month-ahead out-of-sample stock return predictions for each method. We then sort stocks into deciles based on each model's forecasts. We reconstitute portfolios each month using value weights. Finally, we construct a zero-net-investment portfolio that buys the highest expected return stocks (decile 10) and sells the lowest (decile 1).

Table 7 reports results. Out-of-sample portfolio performance aligns very closely with results on machine learning forecast accuracy reported earlier. Realized returns generally increase monotonically with machine learning forecasts from every method (with occasional exceptions, such as decile 8 of NN1). Neural network models again dominate linear models and tree-based approaches. In particular, for all but the most extreme deciles, the quantitative match between predicted returns and average realized returns using neural networks is extraordinarily close. The best 10–1 strategy comes from NN4,

Table 6
Market timing Sharpe ratio gains

	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
A. Common factor portfolios												
S&P 500	0.07	0.05	−0.06	0.12	0.19	0.18	0.19	0.22	0.20	0.26	0.22	0.19
SMB	0.06	0.17	0.09	0.24	0.26	0.00	−0.07	0.21	0.18	0.15	0.09	0.11
HML	0.00	0.01	0.04	−0.03	−0.02	0.04	0.02	0.04	0.06	0.04	0.02	0.01
RMW	0.00	−0.01	−0.06	−0.19	−0.13	−0.11	−0.01	−0.03	−0.09	0.01	0.01	−0.07
CMA	0.02	0.02	0	−0.09	−0.05	0.08	−0.01	0.00	0.01	0.05	0.04	0.06
UMD	0.01	−0.06	−0.02	−0.02	−0.07	−0.04	−0.07	−0.04	−0.08	−0.04	−0.10	−0.01
B. Subcomponents of factor portfolios												
Big value	−0.01	0.06	−0.03	0.09	0.06	0.09	0.08	0.11	0.11	0.13	0.10	0.11
Big growth	0.08	−0.01	−0.08	0.10	0.17	0.20	0.21	0.22	0.20	0.26	0.22	0.21
Big neutral	0.06	0.03	−0.06	0.11	0.16	0.13	0.17	0.23	0.21	0.23	0.23	0.21
Small value	−0.04	0.15	0.09	0.01	0.08	0.07	0.08	0.11	0.11	0.10	0.11	0.13
Small growth	0.00	0.03	−0.06	−0.03	−0.05	0.04	0.05	0.02	0.03	0.03	0.02	0.02
Small neutral	0.02	0.09	0.05	0.03	0.04	0.11	0.11	0.09	0.08	0.10	0.09	0.11
Big conservative	0.08	0.02	−0.04	0.08	0.15	0.09	0.13	0.17	0.14	0.19	0.16	0.14
Big aggressive	0.08	−0.01	−0.11	0.01	0.13	0.22	0.18	0.21	0.19	0.23	0.20	0.20
Big neutral	0.04	−0.01	−0.08	0.09	0.11	0.09	0.11	0.13	0.12	0.18	0.18	0.16
Small conservative	0.04	0.17	0.12	0.02	0.05	0.17	0.15	0.11	0.11	0.14	0.13	0.15
Small aggressive	0.01	0.05	−0.06	−0.05	−0.03	0.08	0.06	0.02	0.05	0.06	0.04	0.05
Small neutral	0.01	0.06	0.03	0.01	0.04	0.08	0.09	0.07	0.06	0.08	0.07	0.09
Big robust	0.10	0.07	−0.07	0.11	0.18	0.17	0.18	0.18	0.16	0.22	0.19	0.16
Big weak	0.05	0.12	0.05	0.09	0.12	0.21	0.17	0.22	0.20	0.21	0.18	0.19
Big neutral	0.09	0.00	−0.04	0.09	0.20	0.19	0.17	0.22	0.21	0.24	0.21	0.20
Small robust	0.09	0.04	−0.03	0.00	0.00	0.10	0.07	0.04	0.05	0.08	0.08	0.08
Small weak	−0.03	0.09	0.00	−0.03	−0.02	0.07	0.07	0.06	0.06	0.06	0.05	0.06
Small neutral	0.04	0.04	−0.03	0.00	0.01	0.11	0.09	0.04	0.04	0.07	0.07	0.08
Big up	0.10	0.05	−0.06	0.10	0.21	0.16	0.14	0.17	0.14	0.17	0.18	0.17
Big down	−0.02	0.09	−0.08	−0.02	0.02	0.08	0.10	0.10	0.07	0.12	0.11	0.09
Big medium	−0.01	0.04	−0.06	0.14	0.09	0.17	0.20	0.22	0.21	0.25	0.22	0.19
Small up	0.08	0.13	0.10	0.05	0.07	0.16	0.12	0.07	0.06	0.08	0.07	0.10
Small down	−0.14	0.04	−0.05	−0.09	−0.05	0.06	0.04	0.01	0.01	0.02	0.01	0.01
Small medium	0.05	0.11	0.07	0.08	0.09	0.13	0.15	0.13	0.12	0.14	0.13	0.15

This table documents improvement in annualized Sharpe ratio ($SR^* - SR$). We compute the SR^* by weighting the portfolios based on a market timing strategy (see Campbell and Thompson 2008).

which returns on average 2.3% per month (27.1% on an annualized basis). Its monthly volatility is 5.8% (20.1% annualized), amounting to an annualized out-of-sample Sharpe ratio of 1.35.

While value-weight portfolios are less sensitive to trading cost considerations, it is perhaps more natural to study equal weights in our analysis because our statistical objective functions minimize equally weighted forecast errors. Table A.9 in the Internet Appendix reports the performance of machine learning portfolios using an equal-weight formation. The qualitative conclusions of this table are identical to those of Table 7, but the Sharpe ratios are substantially higher. For example, the long-short decile spread portfolio based on the NN4 model earns an annualized Sharpe ratio of 2.45 with equal weighting. To identify the extent to which equal-weight results are driven by micro-cap stocks, Table A.10 in the Internet Appendix reports equal-weight portfolio

Table 7
Performance of the machine learning portfolios

	OLS-3+H				PLS				PCR			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low(L)	−0.17	0.40	5.90	0.24	−0.83	0.29	5.31	0.19	−0.68	0.03	5.98	0.02
2	0.17	0.58	4.65	0.43	−0.21	0.55	4.96	0.38	−0.11	0.42	5.25	0.28
3	0.35	0.60	4.43	0.47	0.12	0.64	4.63	0.48	0.19	0.53	4.94	0.37
4	0.49	0.71	4.32	0.57	0.38	0.78	4.30	0.63	0.42	0.68	4.64	0.51
5	0.62	0.79	4.57	0.60	0.61	0.77	4.53	0.59	0.62	0.81	4.66	0.60
6	0.75	0.92	5.03	0.63	0.84	0.88	4.78	0.64	0.81	0.81	4.58	0.61
7	0.88	0.85	5.18	0.57	1.06	0.92	4.89	0.65	1.01	0.87	4.72	0.64
8	1.02	0.86	5.29	0.56	1.32	0.92	5.14	0.62	1.23	1.01	4.77	0.73
9	1.21	1.18	5.47	0.75	1.66	1.15	5.24	0.76	1.52	1.20	4.88	0.86
High(H)	1.51	1.34	5.88	0.79	2.25	1.30	5.85	0.77	2.02	1.25	5.60	0.77
H-L	1.67	0.94	5.33	0.61	3.09	1.02	4.88	0.72	2.70	1.22	4.82	0.88
	ENet+H				GLM+H				RF			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low(L)	−0.04	0.24	5.44	0.15	−0.47	0.08	5.65	0.05	0.29	−0.09	6.00	−0.05
2	0.27	0.56	4.84	0.40	0.01	0.49	4.80	0.35	0.44	0.38	5.02	0.27
3	0.44	0.53	4.50	0.40	0.29	0.65	4.52	0.50	0.53	0.64	4.70	0.48
4	0.59	0.72	4.11	0.61	0.50	0.72	4.59	0.55	0.60	0.60	4.56	0.46
5	0.73	0.72	4.42	0.57	0.68	0.70	4.55	0.53	0.67	0.57	4.51	0.44
6	0.87	0.85	4.60	0.64	0.84	0.84	4.53	0.65	0.73	0.64	4.54	0.49
7	1.01	0.87	4.75	0.64	1.00	0.86	4.82	0.62	0.80	0.67	4.65	0.50
8	1.16	0.88	5.20	0.59	1.18	0.87	5.18	0.58	0.87	1.00	4.91	0.71
9	1.36	0.80	5.61	0.50	1.40	1.04	5.44	0.66	0.96	1.23	5.59	0.76
High(H)	1.66	0.84	6.76	0.43	1.81	1.14	6.33	0.62	1.12	1.53	7.27	0.73
H-L	1.70	0.60	5.37	0.39	2.27	1.06	4.79	0.76	0.83	1.62	5.75	0.98
	GBRT+H				NN1				NN2			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low(L)	−0.45	0.18	5.60	0.11	−0.38	−0.29	7.02	−0.14	−0.23	−0.54	7.83	−0.24
2	−0.16	0.49	4.93	0.35	0.16	0.41	5.89	0.24	0.21	0.36	6.08	0.20
3	0.02	0.59	4.75	0.43	0.44	0.51	5.07	0.35	0.44	0.65	5.07	0.44
4	0.17	0.63	4.68	0.46	0.64	0.70	4.56	0.53	0.59	0.73	4.53	0.56
5	0.34	0.57	4.70	0.42	0.80	0.77	4.37	0.61	0.72	0.81	4.38	0.64
6	0.46	0.77	4.48	0.59	0.95	0.78	4.39	0.62	0.84	0.84	4.51	0.65
7	0.59	0.52	4.73	0.38	1.11	0.81	4.40	0.64	0.97	0.95	4.61	0.71
8	0.72	0.72	4.92	0.51	1.31	0.75	4.86	0.54	1.13	0.93	5.09	0.63
9	0.88	0.99	5.19	0.66	1.58	0.96	5.22	0.64	1.37	1.04	5.69	0.63
High(H)	1.11	1.17	5.88	0.69	2.19	1.52	6.79	0.77	1.99	1.38	6.98	0.69
H-L	1.56	0.99	4.22	0.81	2.57	1.81	5.34	1.17	2.22	1.92	5.75	1.16
	NN3				NN4				NN5			
	Pred	Avg	SD	SR	Pred	Avg	SD	SR	Pred	Avg	SD	SR
Low(L)	−0.03	−0.43	7.73	−0.19	−0.12	−0.52	7.69	−0.23	−0.23	−0.51	7.69	−0.23
2	0.34	0.30	6.38	0.16	0.30	0.33	6.16	0.19	0.23	0.31	6.10	0.17
3	0.51	0.57	5.27	0.37	0.50	0.42	5.18	0.28	0.45	0.54	5.02	0.37
4	0.63	0.66	4.69	0.49	0.62	0.60	4.51	0.46	0.60	0.67	4.47	0.52
5	0.71	0.69	4.41	0.55	0.72	0.69	4.26	0.56	0.73	0.77	4.32	0.62
6	0.79	0.76	4.46	0.59	0.81	0.84	4.46	0.65	0.85	0.86	4.35	0.68
7	0.88	0.99	4.77	0.72	0.90	0.93	4.56	0.70	0.96	0.88	4.76	0.64
8	1.00	1.09	5.47	0.69	1.03	1.08	5.13	0.73	1.11	0.94	5.17	0.63
9	1.21	1.25	5.94	0.73	1.23	1.26	5.93	0.74	1.34	1.02	6.02	0.58
High(H)	1.83	1.69	7.29	0.80	1.89	1.75	7.51	0.81	1.99	1.46	7.40	0.68
H-L	1.86	2.12	6.13	1.20	2.01	2.26	5.80	1.35	2.22	1.97	5.93	1.15

In this table, we report the performance of prediction-sorted portfolios over the 30-year out-of-sample testing period. All stocks are sorted into deciles based on their predicted returns for the next month. Columns "Pred," "Avg," "SD," and "SR" provide the predicted monthly returns for each decile, the average realized monthly returns, their standard deviations, and Sharpe ratios, respectively. All portfolios are value weighted.

Table 8
Drawdowns, turnover, and risk-adjusted performance of machine learning portfolios

	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
Drawdowns and turnover (value weighted)												
Max DD(%)	69.60	41.13	42.17	60.71	37.09	52.27	48.75	61.60	55.29	30.84	51.78	57.52
Max 1M loss(%)	24.72	27.40	18.38	27.40	15.61	26.21	21.83	18.59	37.02	30.84	33.03	38.95
Turnover(%)	58.20	110.87	125.86	151.59	145.26	133.87	143.53	121.02	122.46	123.50	126.81	125.37
Drawdowns and turnover (equally weighted)												
Max DD(%)	84.74	32.35	31.39	33.70	21.01	46.42	37.19	18.25	25.81	17.34	14.72	21.78
Max 1M loss(%)	37.94	32.35	22.33	32.35	15.74	34.63	22.34	12.79	25.81	12.50	9.01	21.78
Turnover(%)	57.24	104.47	118.07	142.78	137.97	120.29	134.24	112.35	112.43	113.76	114.17	114.34
Risk-adjusted performance (value weighted)												
Mean ret.	0.94	1.02	1.22	0.60	1.06	1.62	0.99	1.81	1.92	2.12	2.26	1.97
FF5+Mom α	0.39	0.24	0.62	−0.23	0.38	1.20	0.66	1.20	1.33	1.52	1.76	1.43
$t(\alpha)$	2.76	1.09	2.89	−0.89	1.68	3.95	3.11	4.68	4.74	4.92	6.00	4.71
R^2	78.60	34.95	39.11	28.04	30.78	13.43	20.68	27.67	25.81	20.84	20.47	18.23
IR	0.54	0.21	0.57	−0.17	0.33	0.77	0.61	0.92	0.93	0.96	1.18	0.92
Risk-adjusted performance (equally weighted)												
Mean ret.	1.34	2.08	2.45	2.11	2.31	2.38	2.14	2.91	3.31	3.27	3.33	3.09
FF5+Mom α	0.83	1.40	1.95	1.32	1.79	1.88	1.87	2.60	3.07	3.02	3.08	2.78
$t(\alpha)$	6.64	5.90	9.92	4.77	8.09	6.66	8.19	10.51	11.66	11.70	12.28	10.68
R^2	84.26	26.27	40.50	20.89	21.25	19.91	11.19	13.98	10.60	9.63	11.57	14.54
IR	1.30	1.15	1.94	0.93	1.58	1.30	1.60	2.06	2.28	2.29	2.40	2.09

The top two panels report value-weighted and equally weighted portfolio maximum drawdowns ("Max DD"), the most extreme negative monthly return ("Max 1M Loss"), and the average monthly percentage change in holdings ("Turnover"). The bottom panel reports average monthly returns expressed as a percentage as well as alphas, information ratios (IR), and R^2 with respect to the Fama-French five-factor model augmented to include the momentum factor.

results excluding stocks that fall below the 20th percentile of the NYSE size distribution. In this case, the NN4 long-short decile spread earns a Sharpe ratio of 1.69.

As recommended by Lewellen (2015), the OLS-3 model is an especially parsimonious and robust benchmark model. He also recommends somewhat larger OLS benchmark models with either 7 or 15 predictors, which we report in Table A.11 in the Internet Appendix. The larger OLS models improve over OLS-3, but are nonetheless handily outperformed by tree-based models and neural networks.

The top panel of Table 8 reports drawdowns, portfolio turnover, and risk-adjusted performance of 10–1 spread portfolios from each machine learning model, for both value and equally weighted strategies. We define maximum drawdown of a strategy as

$$\text{MaxDD} = \max_{0 \le t_1 \le t_2 \le T} (Y_{t_1} - Y_{t_2})$$

where Y_t is the cumulative log return from date 0 through t.

Not only do neural network portfolios have higher Sharpe ratios than alternatives, they also have comparatively small drawdowns, particularly for equal-weight portfolios. The maximum drawdown experienced for the NN4

strategy is 51.8% and 14.7% for value and equal weights, respectively. In contrast, the maximum drawdown for OLS-3 are 69.6% and 84.7%, respectively. Equal-weight neural network strategies also experience the most mild 1-month losses. For example, for NN4, the worst 1-month performance is a −9.01% return, which is the least extreme monthly loss among all models with either weighting scheme.

We define the strategy's average monthly turnover as

$$\text{Turnover} = \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i}\left|w_{i,t+1} - \frac{w_{i,t}(1+r_{i,t+1})}{1+\sum_j w_{j,t}r_{j,t+1}}\right|\right),$$

where $w_{i,t}$ is the weight of stock i in the portfolio at time t. For NN1 through NN5, turnover is consistently between 110% and 130% per month, and turnover for tree-based methods and penalized regression is slightly higher. As a frame of reference, the monthly turnover of a short-term reversal decile spread is 172.6% per month while for a size decile spread it is 22.9%. Given the large role of price trend predictors selected by all machine learning approaches, it is perhaps unsurprising that their outperformance is accomplished with comparatively high portfolio turnover.

The bottom panel of Table 8 reports risk-adjusted performance of machine learning portfolios based on factor pricing models. In a linear factor model, the tangency portfolio of the factors themselves represents the maximum Sharpe ratio portfolio in the economy. Any portfolio with a higher Sharpe ratio than the factor tangency portfolio possesses alpha with respect to the model. From prior work, the out-of-sample factor tangency portfolios of the Fama-French three and five-factor models have Sharpe ratios of roughly 0.8 and 1.3, respectively (Kelly, Pruitt, and Su 2019). It is unsurprising then that portfolios formed on the basis of machine learning forecasts earn large and significant alphas versus these models. A six-factor model (that appends a momentum factor to the Fama-French five-factor model) explains as much as 40% of the average return for strategies based on linear models, but explains only about 10% to 30% of variation in neural network-based portfolios. As a result, neural networks have information ratios ranging from 0.9 to 2.4 depending on the number of layers and the portfolio weighting scheme. Test statistics for the associated alphas are highly significant for both tree models and all neural network models.

The results of Table 8 are visually summarized in Figure 9, which reports cumulative performance for the long and short sides for select strategies, along with the cumulative market excess return as a benchmark. NN4 dominates the other models by a large margin in both directions. Interestingly, the short side of all portfolios is essentially flat in the post-2000 sample.[39]

Finally, we consider two metastrategies that combine forecasts of all machine learning portfolios. The first is a simple equally weighted average of decile

[39] Figure A.7 in the Internet Appendix plots the corresponding cumulative returns for equally weighted strategies.

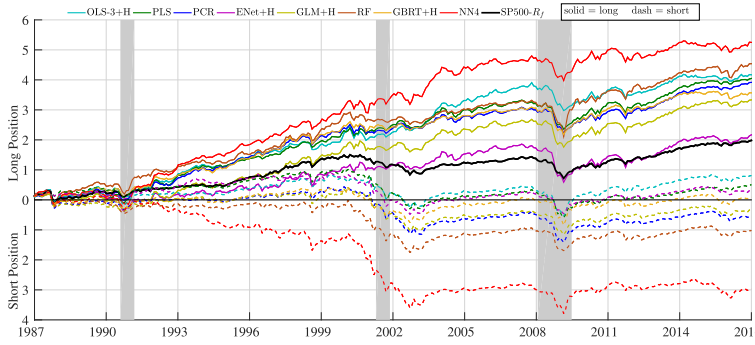


Figure 9
Cumulative return of machine learning portfolios
The figure shows the cumulative log returns of portfolios sorted on out-of-sample machine learning return forecasts. The solid and dashed lines represent long (top decile) and short (bottom decile) positions, respectively. The shaded periods show NBER recession dates. All portfolios are value weighted.

long-short portfolios from our eleven machine learning methods. This achieves a stock-level predictive R^2 of 0.43% per month and an equally weighted decile spread Sharpe ratio of 2.49, both of which are higher than any single method on its own. The value weighted decile spread Sharpe ratio is 1.33, which is slightly lower than that for the NN4 model.

The second metastrategy rotates between machine learning methods by selecting the best machine learning model for each 1-year test sample based on the predictive R^2 during the corresponding validation sample. Over our 30-year out-of-sample period, this method selects NN3 11 times, NN1 7 times, GBRT 6 times, NN2 5 times, and NN4 1 time. This method delivers the highest overall panel R^2 (0.45%), but underperforms the standalone NN4 model in terms of decile spread Sharpe ratio (2.42 and 1.23 with equal and value weights, respectively). Interestingly, the meta-strategy's predictive R^2 gain is statistically insignificant relative to the best standalone neural network models.

3. Conclusion

Using the empirical context of return prediction as a proving ground, we perform a comparative analysis of methods in the machine learning repertoire. At the highest level, our findings demonstrate that machine learning methods can help improve our empirical understanding of asset prices. Neural networks and, to a lesser extent, regression trees, are the best performing methods. We track down the source of their predictive advantage to accommodation of nonlinear interactions that are missed by other methods. We also find that "shallow" learning outperforms "deep" learning, which differs from the typical conclusion in other fields, such as computer vision or bioinformatics, and is likely due to the comparative dearth of data and low signal-to-noise ratio in asset pricing problems. Machine learning methods are most valuable for

forecasting larger and more liquid stock returns and portfolios. Lastly, we find that *all* methods agree on a fairly small set of dominant predictive signals, the most powerful predictors being associated with price trends including return reversal and momentum. The next most powerful predictors are measures of stock liquidity, stock volatility, and valuation ratios.

The overall success of machine learning algorithms for return prediction brings promise for both economic modeling and for practical aspects of portfolio choice. With better measurement through machine learning, risk premiums are less shrouded in approximation and estimation error, thus the challenge of identifying reliable economic mechanisms behind asset pricing phenomena becomes less steep. Finally, our findings help justify the growing role of machine learning throughout the architecture of the burgeoning fintech industry.

References

Bishop, C. M. 1995. *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Box, G. E. P. 1953. Non-normality and tests on variances. *Biometrika* 40:318–35.

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

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen. 1984. *Classification and regression trees*. Boca, Raton, FL: CRC press.

Butaru, F., Q. Chen, B. Clark, S. Das, A. W. Lo, and A. Siddique. 2016. Risk and risk management in the credit card industry. *Journal of Banking & Finance* 72:218–39.

Campbell, J. Y., and S. B. Thompson. 2008. Predicting excess stock returns out of sample: Can anything beat the historical average? *Review of Financial Studies* 21:1509–31.

Cochrane, J. H. 2007. The dog that did not bark: A defense of return predictability. *Review of Financial Studies* 21:1533–75.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2:303–14.

de Jong, Sijmen. 1993. SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems* 18:251–63. http://www.sciencedirect.com/science/article/pii/016974399385002X.

Diebold, F. X. 2015. Comparing Predictive Accuracy, Twenty Years Later: A Personal Perspective on the Use and Abuse of Diebold-Mariano Tests. *Journal of Business & Economic Statistics* 33:1–9.

Diebold, F. X., and R. S. Mariano. 1995. Comparing Predictive Accuracy. *Journal of Business & Economic Statistics* 13:134–44.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *International workshop on multiple classier systems*, eds. F. Schwenker, F. Roli, and J. Kittler, 1–15. New York: Springer.

Dimopoulos, Y., P. Bourret, and S. Lek. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters* 2:1–4.

Eldan, R., and O. Shamir. 2016. The power of depth for feedforward neural networks. In *29th Annual Conference on Learning Theory*, eds. V. Feldman, A. Rakhlin, and O. Shamir, 907–40. Brookline, MA: Microtome Publishing.

Fama, E. F., and K. R. French. 1993. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics* 33:3–56.

———. 2008. Dissecting anomalies. *Journal of Finance* 63:1653–78.

———. 2015. A five-factor asset pricing model. *Journal of Financial Economics* 116:1–22. http://dx.doi.org/10.1016/j.jfineco.2014.10.010.

Fan, J., Q. Li, and Y. Wang. 2017. Estimation of high dimensional mean regression in the absence of symmetry and light tail assumptions. *Journal of the Royal Statistical Society, Series B* 79:247–65.

Feng, G., S. Giglio, and D. Xiu. Forthcoming. Taming the factor zoo: A test of new factors. *Journal of Finance*.

Freund, Y. 1995. Boosting a Weak Learning Algorithm by Majority. *Information and Computation* 121:256–85.

Freyberger, J., A. Neuhierl, and M. Weber. 2020. Dissecting characteristics nonparametrically. *Review of Financial Studies* 33: 2326–77.

Friedman, J. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 5:1189–232.

Friedman, J., T. Hastie, and R. Tibshirani. 2000. Additive logistic regression: A statistical view of boosting. *Annals of Statistics* 28:337–74.

Giglio, S. W., and D. Xiu. 2016. Asset Pricing with Omitted Factors. Tech. Report, University of Chicago.

Glorot, X., A. Bordes, and Y. Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15, 315–323. Brookline, MA: Microtome Publishing.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep learning*. Cambridge: MIT Press.

Green, J., J. R. M. Hand, and X. F. Zhang. 2013. The supraview of return predictive signals. *Review of Accounting Studies* 18:692–730.

———. 2017. The characteristics that provide independent information about average us monthly stock returns. *Review of Financial Studies* 30:4389–436.

Gu, S., B. T. Kelly, and D. Xiu. 2019. Autoencoder asset pricing models. Working Paper, Yale University.

Hansen, L. K., and P. Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12:993–1001.

Harvey, C. R., and W. E. Ferson. 1999. Conditioning variables and the cross-section of stock returns. *Journal of Finance* 54:1325–60.

Harvey, C. R., and Y. Liu. 2016. Lucky factors. Technical Report, Duke University.

Harvey, C. R., Y. Liu, and H. Zhu. 2016. ... And the cross-section of expected returns. *Review of Financial Studies* 29:5–68.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The elements of statistical learning*. New York: Springer.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, 770–78. Piscataway, NJ: IEEE Publications.

Heaton, J. B., N. G. Polson, and J. H. Witte. 2016. Deep learning in finance. Preprint, https://arxiv.org/abs/1602.06561.

Hinton, G. E., S. Osindero, and Y.-W. Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18:1527–54.

Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2:359–66.

Huber, P. J. 1964. Robust estimation of a location parameter. *Annals of Mathematical Statistics* 35:73–101.

Hutchinson, J. M., A. W. Lo, and T. Poggio. 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance* 49:851–89.

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 International Conference on Machine Learning*, eds. F. Bach and D. Blei, 448–56. New York: ACM.

Jaggi, M. 2013. An equivalence between the lasso and support vector machines. In *Regularization, optimization, kernels, and support vector machines*, eds. J. A. K. Suykens, M. Signoretto, and A. Argyriou, 1–26. Boca Raton, FL: CRC Press.

Jarrett, K., K. Kavukcuoglu, M.'A. Ranzato, and Y. Lecun. 2009. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, 2146–53. Piscataway, NJ: IEEE.

Kelly, B., and S. Pruitt. 2013. Market expectations in the cross-section of present values. *Journal of Finance* 68:1721–56.

———. 2015. The three-pass regression filter: A new approach to forecasting using many predictors. *Journal of Econometrics* 186:294–316.

Kelly, B., S. Pruitt, and Y. Su. 2019. Characteristics *are* covariances: A unified model of risk and return. *Journal of Financial Economics*.

Khandani, A. E., A. J. Kim, and A. W. Lo. 2010. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance* 34:2767–87.

Kingma, D., and J. Ba. 2014. Adam: A method for stochastic optimization. Preprint, https://arxiv.org/abs/1412.6980.

Koijen, R., and S. V. Nieuwerburgh. 2011. Predictability of Returns and Cash Flows. *Annual Review of Financial Economics* 3:467–91.

Kozak, S. 2019. Kernel trick for the cross section. Working Paper, University of Maryland.

Kozak, S., S. Nagel, and S. Santosh. 2020. Shrinking the cross section. *Journal of Financial Economics* 135:271–92.

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

Lo, A. W., and A. C. MacKinlay. 1990. Data-snooping biases in tests of financial asset pricing models. *Review of Financial Studies* 3:431–67.

Masters, T. 1993. *Practical neural network recipes in C++*. New York: Academic Press.

Moritz, B., and T. Zimmermann. 2016. Tree-based conditional portfolio sorts: The relation between past and future stock returns. Working Paper, Ludwig Maximilian University of Munich.

Nair, V., and G. E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, eds. J. Fürnkranz and T. Joachims, 807–14. New York: ACM.

Rapach, D., and G. Zhou. 2013. Forecasting stock returns. In *Handbook of economic forecasting*, eds. G. Elliott and A. Timmermann, 328–83. Amsterdam, the Netherlands: Elsevier.

Rapach, D. E., J. K. Strauss, and G. Zhou. 2013. International stock return predictability: What is the role of the United States? *Journal of Finance* 68:1633–62.

Rosenberg, B. 1974. Extra-Market Components of Covariance in Security Returns. *Journal of Financial and Quantitative Analysis* 9:263–74.

Schapire, R. E. 1990. The Strength of Weak Learnability. *Machine Learning* 5:197–227.

Sirignano, J., A. Sadhwani, and K. Giesecke. 2016. Deep Learning for Mortgage Risk. Working Paper, University of Illinois, Urbana-Champaign.

Tukey, J. W. 1960. A survey of sampling from contaminated distributions. In *Contributions to probability and statistics*, eds. I. Olkin, S. G. Ghurye, W. Hoeding, W. G. Madow, and H. B. Mann. Stanford, CA: Stanford University Press.

Welch, I., and A. Goyal. 2008. A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies* 21:1455–508.

White, H. 1980. Using least squares to approximate unknown regression functions. *International Economic Review* 21:149–70.

Wilson, D. R., and T. R. Martinez. 2003. The General Inefficiency of Batch Training for Gradient Descent Learning. *Neural Networks* 16:1429–51.

Yao, J., Y. Li, and C. L. Tan. 2000. Option price forecasting using neural networks. *Omega* 28:455–66.