THE UNIVERSITY OF CHICAGO

MACHINE LEARNING IN EMPIRICAL ASSET PRICING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE UNIVERSITY OF CHICAGO
BOOTH SCHOOL OF BUSINESS
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

BY
SHIHAO GU

CHICAGO, ILLINOIS
DECEMBER 2021

The dissertation is dedicated to my parents Biwu Gu and Yunzhu Han, and my wife Wenting Feng, without whose unconditional love and support I would never have been able to reach the current stage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This journey would not have been possible without the support of my family, my friends, my professors and especially my advisors.

I would like to express my sincere gratitude to my supervisor Prof. Dacheng Xiu for his noble guidance with full encouragement and enthusiasm. I have benefited greatly not only from his wealth of knowledge, but also his earnest attitude towards research and life. I am grateful to my another advisor and co-auther Prof. Bryan Kelly, for his invaluable guidance and immense knowledge on our research projects. I am also grateful to the rest of my committee, Prof. Tengyuan Liang, and Prof. Nicholas Polson, for their great efforts and support to my dissertation writing and revision.

I must thank my families. To my parents, Biwu Gu and Yunzhu Han, thank you for encouraging and supporting me to go abroad and pursue my Ph.D. degree in ChicagoBooth. To my wife, Wenting Feng, thank you for marrying me. Meeting you in our high school was the best thing ever happened to me. Thank you for providing me with the moral and emotional support along the way.

I would like to thank my fellow doctoral students for their feedback, cooperation, and the lifetime friendship. Also, I would like to express my gratitude to staffs of the Ph.D. program. Last but not least, I am grateful to all the people met in Chicago, who are the best scenery through this journey.

CHAPTER 1
EMPIRICAL ASSET PRICING VIA MACHINE LEARNING

Research in collaboration with Bryan Kelly and Dacheng Xiu. A revised version of this paper is published on The Review of Financial Studies.

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)

1.1 Introduction

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 the 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

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 et al. [2019b] and Kelly et al. [2019].

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 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

2. Green et al. [2013] count 330 stock-level predictive signals in published or circulated drafts. Harvey et al. [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.

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 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

3. See, for example, Hastie et al. [2009].

4. See, for example, Jaggi [2013] and Hastie et al. [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].

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 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 general-

ized 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 1.5.1, 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 et al., 2019, Gu et al., 2019b, Feng et al., 2019a], 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 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 et al. [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 et al., 1994, Yao et al., 2000, among others]. Khandani et al. [2010] and Butaru et al. [2016] use regression trees to predict consumer credit card delinquencies and defaults. Sirignano et al. [2016] estimate a deep neural network for mortgage prepayment, delinquency, and foreclosure. Heaton et al. [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 et al. [2019] use dimension reduction methods to estimate and test factor pricing models. Moritz and Zimmermann [2016] apply tree-based models to portfolio sorting. Kozak et al. [2019] and Freyberger et al. [2019] 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.

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.

1.2 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 1.5.2 and refer readers to original sources for further background. Internet Appendix 1.5.3 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:

$$r_{i,t+1} = \mathrm{E}_t(r_{i,t+1}) + \epsilon_{i,t+1}, \tag{1.1}$$

where

$$\mathrm{E}_t(r_{i,t+1}) = g^\star(z_{i,t}). \tag{1.2}$$

Stocks are indexed as $i = 1, ..., N_t$ and months by $t = 1, ..., T$. For ease of presentation, we assume a balanced panel of stocks, and defer the discussion on missing data to Section 1.3.1. 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.

1.2.1 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,"

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.

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 1.5.4 provides further details for our sample splitting scheme, and Internet Appendix 1.5.5 summarizes the hyperparameter tuning schemes for each model.

1.2.2 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 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,

$$g(z_{i,t};\theta) = z_{i,t}'\theta. \tag{1.3}$$

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:

$$\mathcal{L}(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(r_{i,t+1} - g(z_{i,t};\theta)\right)^2. \tag{1.4}$$

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.

Extension: Robust objective functions.

In some cases, replacing Equation (1.4) with a weighted least squares objective, such as

$$\mathcal{L}_W(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T} w_{i,t}\left(r_{i,t+1} - g(z_{i,t};\theta)\right)^2, \tag{1.5}$$

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 (1.4) places extreme emphasis on large errors, thus outliers can undermine the stability of OLS-based 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 defined as

$$\mathcal{L}_H(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T} H\left(r_{i,t+1} - g(z_{i,t};\theta), \xi\right), \tag{1.6}$$

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.

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.

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

9. OLS is a special case of the (1.6) 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.

1.2.3 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 (1.3). That is, it continues to consider only the baseline, untransformed predictors. Penalized methods differ by appending a penalty to the original loss function:

$$\mathcal{L}(\theta;\cdot) = \mathcal{L}(\theta) + \phi(\theta;\cdot). \tag{1.7}$$

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

$$\phi(\theta;\lambda,\rho) = \lambda(1-\rho)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2. \tag{1.8}$$

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.

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.

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 1.5.2 for more details).

1.2.4 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

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

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 (1.1)–(1.3). In particular, we reorganize the linear regression $r_{i,t+1} = z'_{i,t}\theta + \epsilon_{i,t+1}$ as

$$R = Z\theta + E, \tag{1.9}$$

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

$$R = (Z\Omega_K)\theta_K + \tilde{E}. \tag{1.10}$$

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 $P \times 1$.

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

$$w_j = \arg\max_w \text{Var}(Zw), \quad \text{s.t.} \quad w'w = 1, \quad \text{Cov}(Zw, Zw_l) = 0, \quad l = 1, 2, \ldots, j-1. \tag{1.11}$$

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 (1.11) 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 jth PLS

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 $\text{Var}(a) := \text{Cov}(a, a)$.

component solve

$$w_j = \arg\max_{w} \ \mathrm{Cov}^2(R, Zw), \quad \text{s.t.} \quad w'w = 1, \quad \mathrm{Cov}(Zw, Zw_l) = 0, \quad l = 1, 2, \ldots, j-1. \tag{1.12}$$

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 (1.12) 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.

1.2.5 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 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 gener-

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

alized 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 1.2.2 and 1.2.3.

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

$$g(z;\theta,p(\cdot)) = \sum_{j=1}^{P} p(z_j)'\theta_j, \tag{1.13}$$

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: $\big(1, z, (z-c_1)^2, (z-c_2)^2, \ldots, (z-c_{K-2})^2\big)$, 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 1.2.2. 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

$$\phi(\theta;\lambda,K) = \lambda \sum_{j=1}^{P} \left(\sum_{k=1}^{K} \theta_{j,k}^2 \right)^{1/2}. \tag{1.14}$$

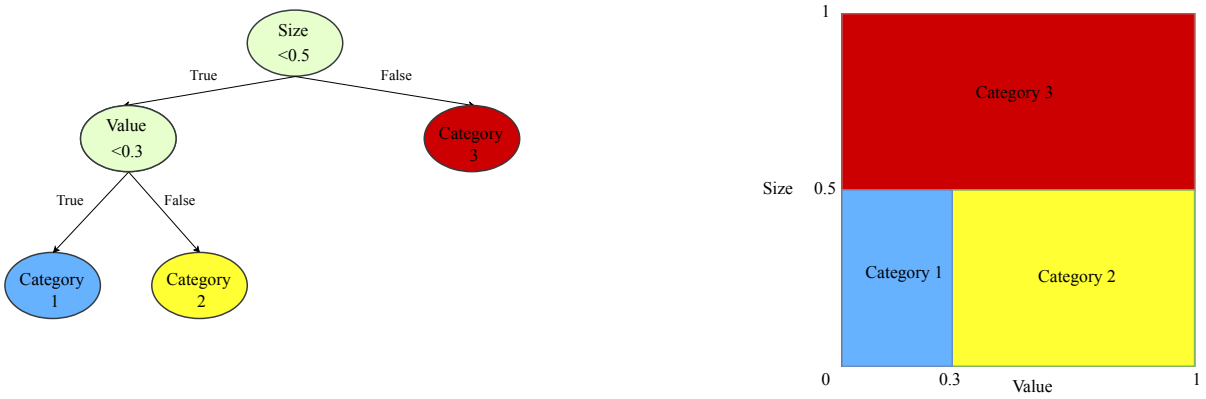
As its name suggests, the group lasso selects either all K spline terms associated with a given characteristic or none of them. We embed this penalty in the general objective of Equation (1.7). Group lasso accommodates either least squares or robust Huber objective, and it uses the same accelerated proximal gradient descent as the elastic net. It has two tuning parameters, λ and K.[14]

1.2.6 Boosted regression trees and random forests

The model in (1.13) 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

14. For additional details, see Internet Appendix 1.5.2. Freyberger et al. [2019] offers a similar model but in the return prediction context.

Figure 1.1: 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.

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 1.1 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.

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.

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

$$g(z_{i,t};\theta,K,L)=\sum_{k=1}^{K}\theta_k\mathbf{1}_{\{z_{i,t}\in C_k(L)\}}, \tag{1.15}$$

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 1.1 has the following prediction equation:

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

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 1.5.2. 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:

$$H(\theta,C)=\frac{1}{|C|}\sum_{z_{i,t}\in C}(r_{i,t+1}-\theta)^2, \tag{1.16}$$

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.

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

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.

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 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 1.5.2. 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

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. [2000a] and Friedman [2001] extend boosting to contexts beyond classification, eventually leading to the gradient boosted regression tree.

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 "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.

1.2.7 Neural networks

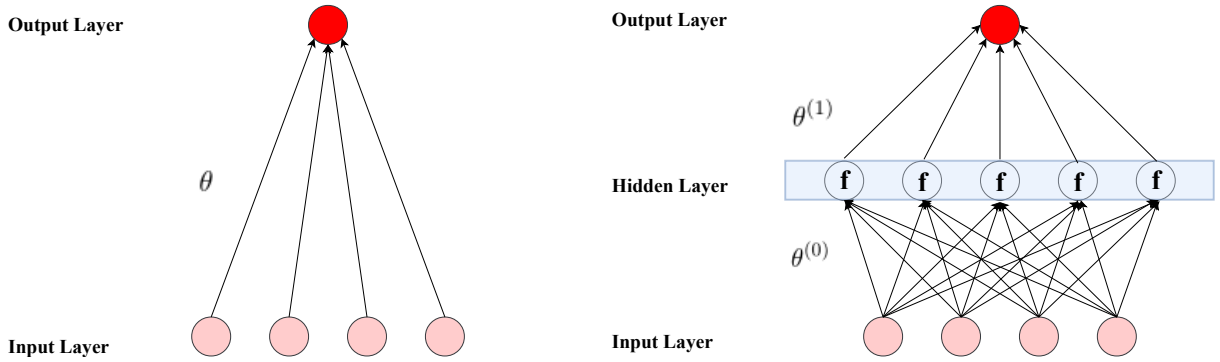
The final nonlinear method that we analyze is the artificial neural network. Arguably the most powerful modeling device in machine learning, neural networks have theoretical underpinnings as "universal approximators" for any smooth predictive association [Hornik et al., 1989, Cybenko, 1989]. They are 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 1.2 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 1.2 shows an example with one hidden layer that contains five neurons. Each neuron draws information linearly from all of

Figure 1.2: 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.

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 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

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 et al. [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. [2016a]).

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.

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

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 et al. [2011].

neuron in layer $l > 0$ is then

$$x_k^{(l)} = \text{ReLU}\left(x^{(l-1)'}\theta_k^{(l-1)}\right), \tag{1.17}$$

with final output

$$g(z;\theta) = x^{(L-1)'}\theta^{(L-1)}. \tag{1.18}$$

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 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 1.5.2).[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

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].

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). 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 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]

1.2.8 Performance evaluation

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

$$R^2_{\text{oos}} = 1 - \frac{\sum_{(i,t)\in\mathcal{T}_3}(r_{i,t+1} - \widehat{r}_{i,t+1})^2}{\sum_{(i,t)\in\mathcal{T}_3} r^2_{i,t+1}}, \tag{1.19}$$

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 et al. [2016].

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

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 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

$$d_{12,t+1} = \frac{1}{n_{3,t+1}} \sum_{i=1}^{n_3} \left(\left(\widehat{e}^{(1)}_{i,t+1} \right)^2 - \left(\widehat{e}^{(2)}_{i,t+1} \right)^2 \right), \tag{1.20}$$

$\widehat{e}^{(1)}_{i,t+1}$ and $\widehat{e}^{(2)}_{i,t+1}$ 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.

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."

1.2.9 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 et al., 2019]. The second, proposed in the neural networks literature by Dimopoulos et al. [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]

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.

1.3 An Empirical Study of U.S. Equities

1.3.1 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

27. In particular, SSD defines the jth variable importance as

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

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 et al. [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].

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

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 1.14 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 (1.2). Throughout our analysis, we define the baseline set of stock-level covariates $z_{i,t}$ as

$$z_{i,t} = x_t \otimes c_{i,t}, \tag{1.21}$$

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 (1.2) and (1.21) 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

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 et al. [2019] and Freyberger et al. [2019].

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.

conditional Euler equation,

$$\mathrm{E}_t(r_{i,t+1}) = \beta'_{i,t}\lambda_t. \tag{1.22}$$

The structure of our feature set in (1.21) 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 (1.22) becomes

$$g^\star(z_{i,t}) = \mathrm{E}_t(r_{i,t+1}) = \beta'_{i,t}\lambda_t = c'_{i,t}\theta'_1\theta_2 x_t = \left(x_t \otimes c_{i,t}\right)' \mathrm{vec}(\theta'_1\theta_2) =: z'_{i,t}\theta, \tag{1.23}$$

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]

1.3.2 The cross-section of individual stocks

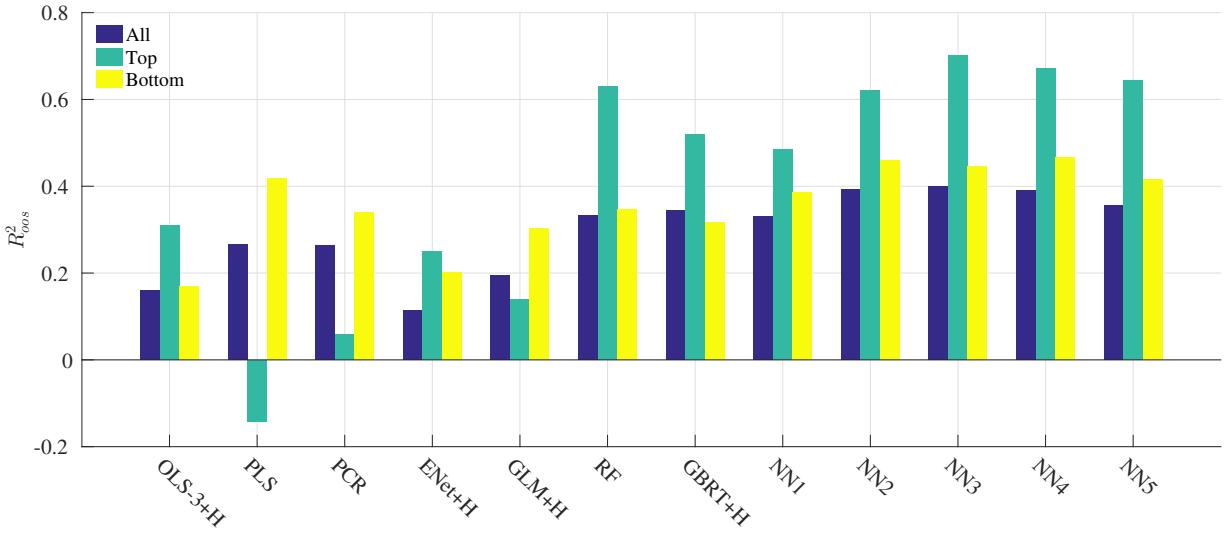
Table 1.1 presents the comparison of machine learning techniques in terms of their out-of-sample predictive R^2. We compare thirteen models in total, including OLS with all covariates, OLS-3 (which preselects size, book-to-market, and momentum as the only covariates), PLS, PCR, elastic net (ENet), generalized 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 1.1 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

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

Table 1.1: 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
New Version	-3.38	0.16	0.26	0.26	0.12	0.20	0.34	0.34	0.32	0.39	0.40	0.39	0.35
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

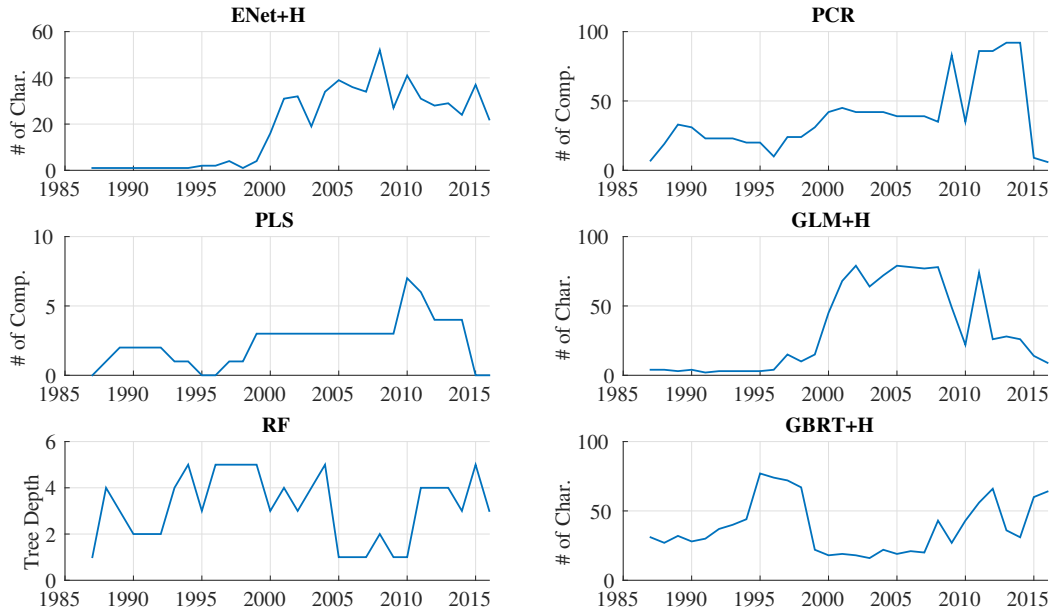


In this table, we report monthly R^2_{oos} for the entire panel of stocks using OLS with all variables (OLS), OLS using only size, book-to-market, and momentum (OLS-3), PLS, PCR, elastic net (ENet), generalize linear model (GLM), random forest (RF), gradient boosted regression trees (GBRT), and neural 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). The new version line reports the results using the updated dataset.

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 1.3 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 especially PCR, raise the out-of-sample R^2 to 0.27% and 0.26%, respectively. Figure 1.3 shows that PCR typically uses

Figure 1.3: 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.

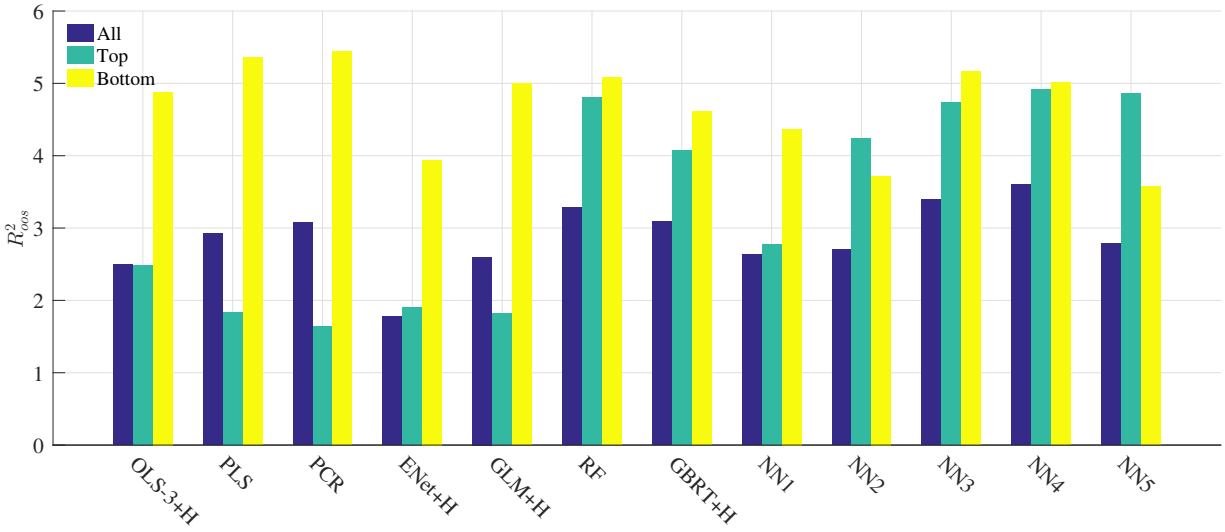
20 to 40 components in its forecasts. PLS, on the other hand, fails to find a single reliable component for much of the early sample, but eventually settles on three to six components. The improvement of dimension reduction over variable selection via elastic net suggests that characteristics are partially 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

Table 1.2: 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
New Version	-33.58	2.46	2.94	3.01	1.75	2.65	3.33	3.07	2.68	2.72	3.42	3.58	2.82
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 1.1).

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 1.1 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

33. Because we hold the five neural networks architectures fixed and simply compare across them, we do not describe their estimated complexity in Figure 1.3.

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 1.2 conducts our analysis at the annual horizon. The comparative performance across different methods is similar to the monthly results shown in Table 1.1, 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 1.1 offers a quantitative comparison of models' predictive performance, Table 1.3 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 1.3 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 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 1.3 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

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.

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

Table 1.3: 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	**3.29***	**3.33***	**3.34***	**3.28***	**3.27***	**3.30***	**3.30***	**3.34***	**3.42***	**3.38***	**3.38***	**3.38***
OLS-3+H		1.42	**1.88**	-0.26	0.62	**1.63**	1.28	1.26	**2.15**	**2.14**	**2.34**	**2.12**
PLS			-0.19	-1.17	-1.46	0.86	0.66	0.62	1.31	1.39	**1.68**	1.07
PCR				-1.09	-1.36	0.85	0.75	0.57	1.16	1.20	1.35	1.02
ENet+H					0.64	**1.91**	1.41	**1.72**	**1.97**	**2.06**	**1.99**	**1.88**
GLM+H						**1.75**	1.21	1.29	**2.29**	**2.18**	**2.66***	**2.35**
RF							0.07	-0.03	0.32	0.38	0.33	0.00
GBRT+H								-0.06	0.16	0.21	0.17	-0.04
NN1									0.55	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.

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.

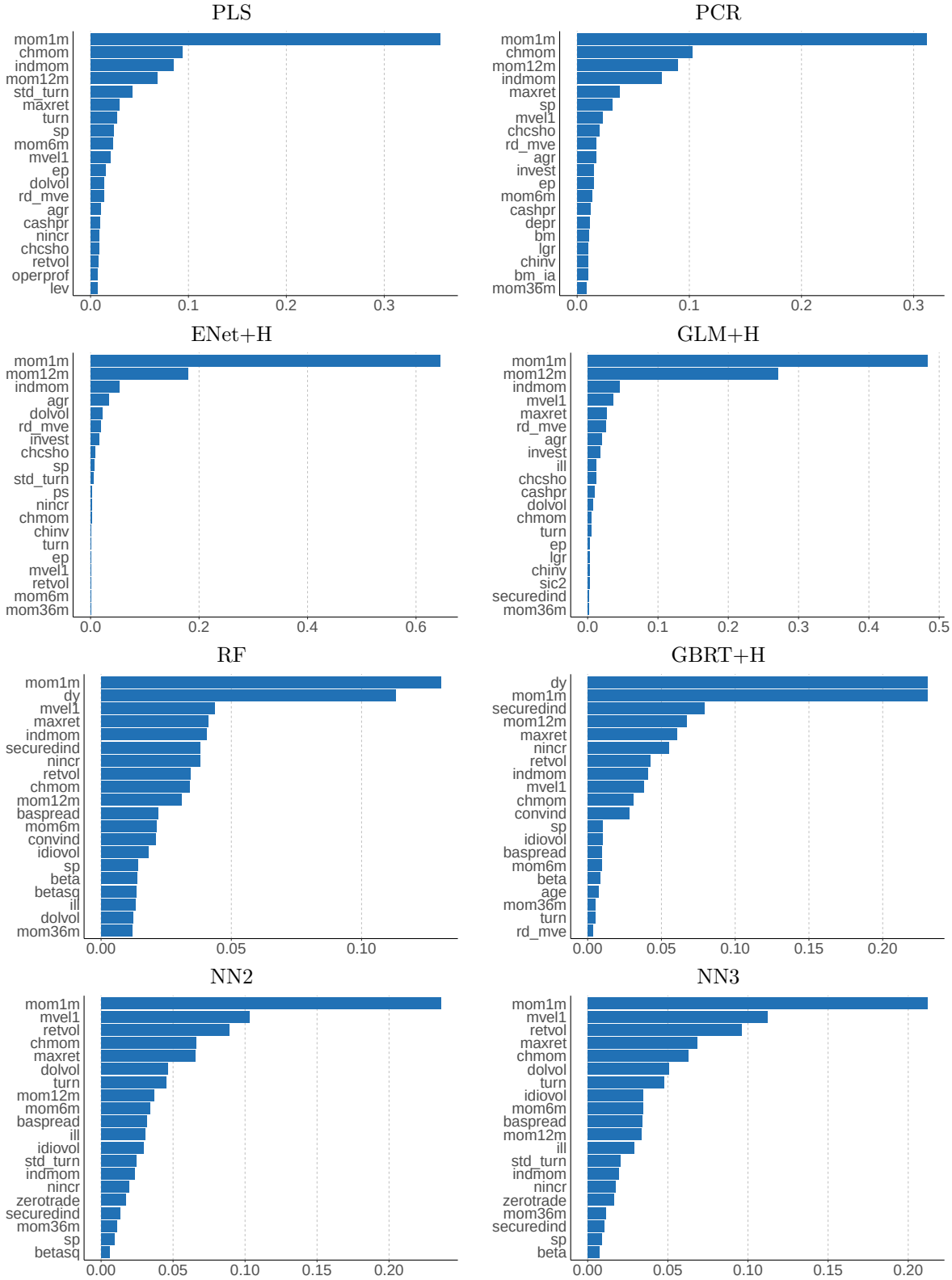
1.3.3 Which covariates matter?

We now investigate the relative importance of individual covariates for the performance of each model using the importance measures described in Section 1.2.9. 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 1.4 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 1.5 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

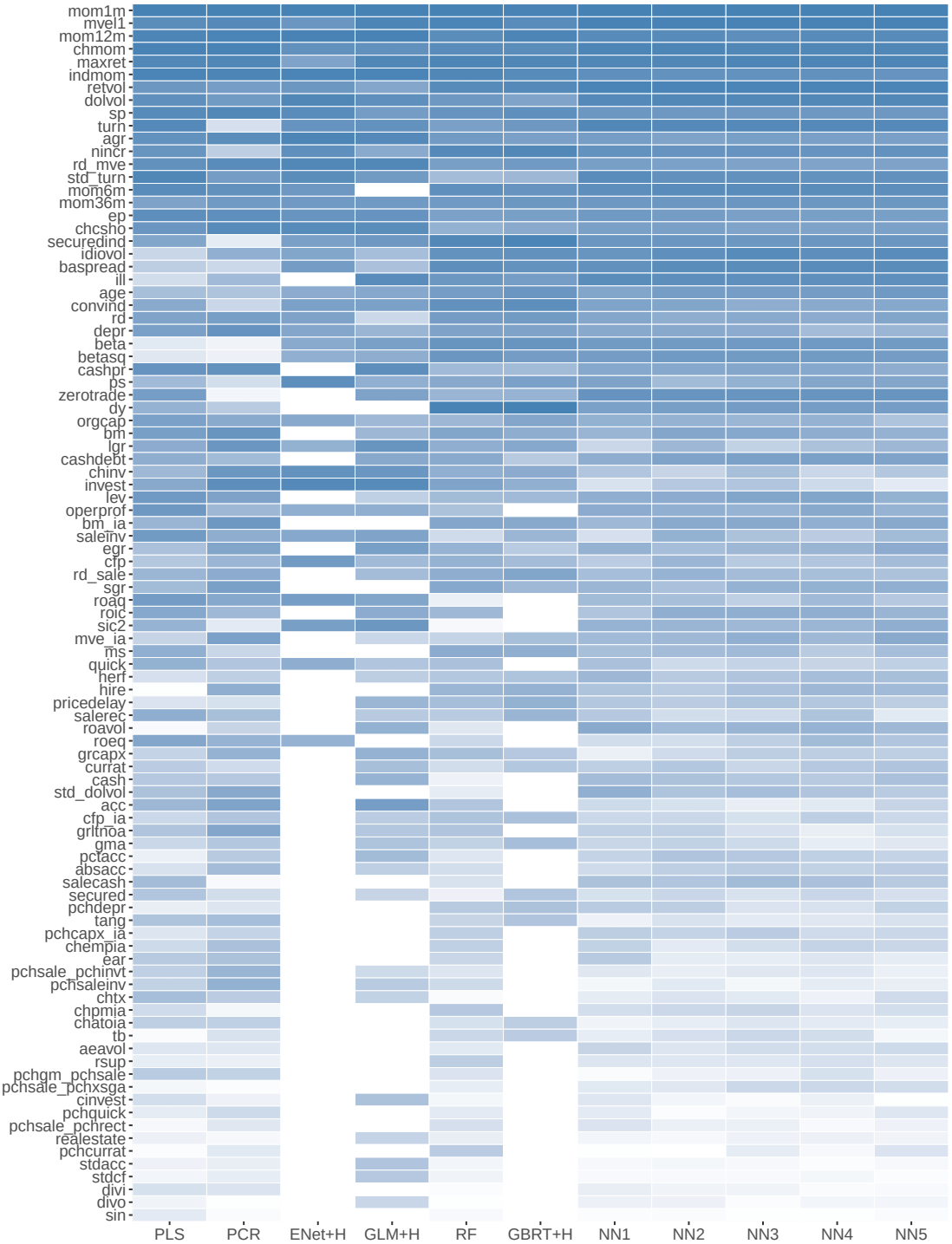
distributed. The statistically significant t-statistics in our analyses do not appear random, but instead follow a pattern in which nonlinear models outperform linear ones.

Figure 1.4: 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.

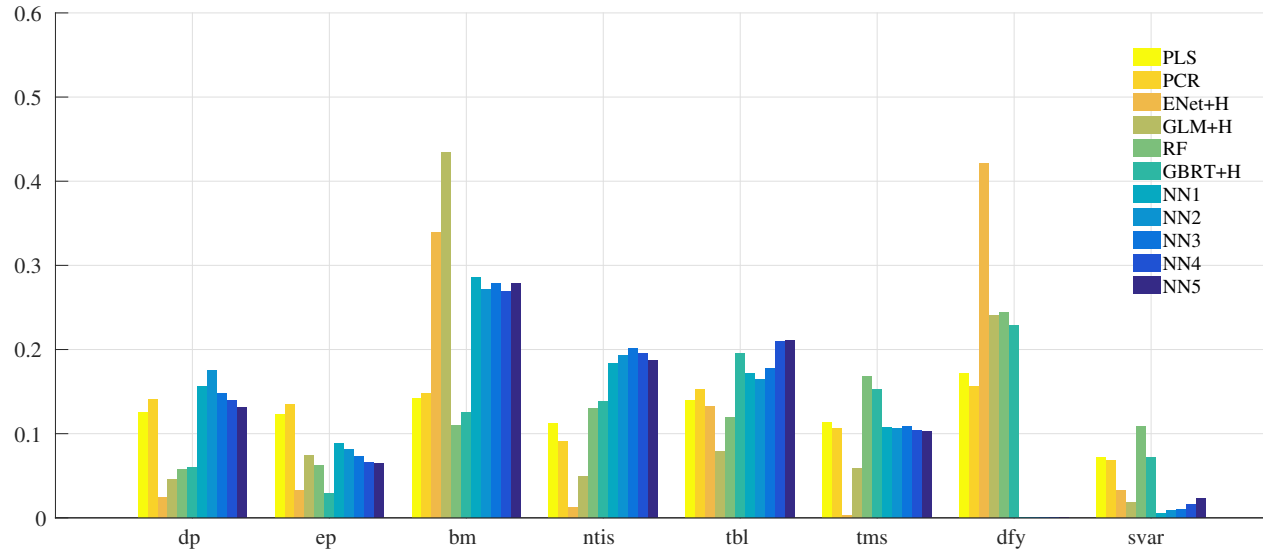
Figure 1.5: Characteristic importance



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.

Table 1.4: 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.

least to most important (lightest to darkest).[36]

Figures 1.4 and 1.5 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 1.5: 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

36. Figure 1.5 is based on the average rank of over thirty recursing training samples. Figure 1.10 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.

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 recent earnings increases (nincr). Figure 1.4 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 et al. [1995], produces very similar results to the 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 1.11 in the Internet Appendix shows the full set of SSD results.

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 (1.24) in Internet Appendix 1.5.1. The parameters are calibrated to have similar behavior as our characteristics data set but are independent of future returns by construction. Figure 1.12 in Internet Appendix 1.5.6 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 1.4 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 1.5. To investigate this, Figure 1.15 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.

Marginal association between characteristics and expected returns.

Figure 1.6 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).

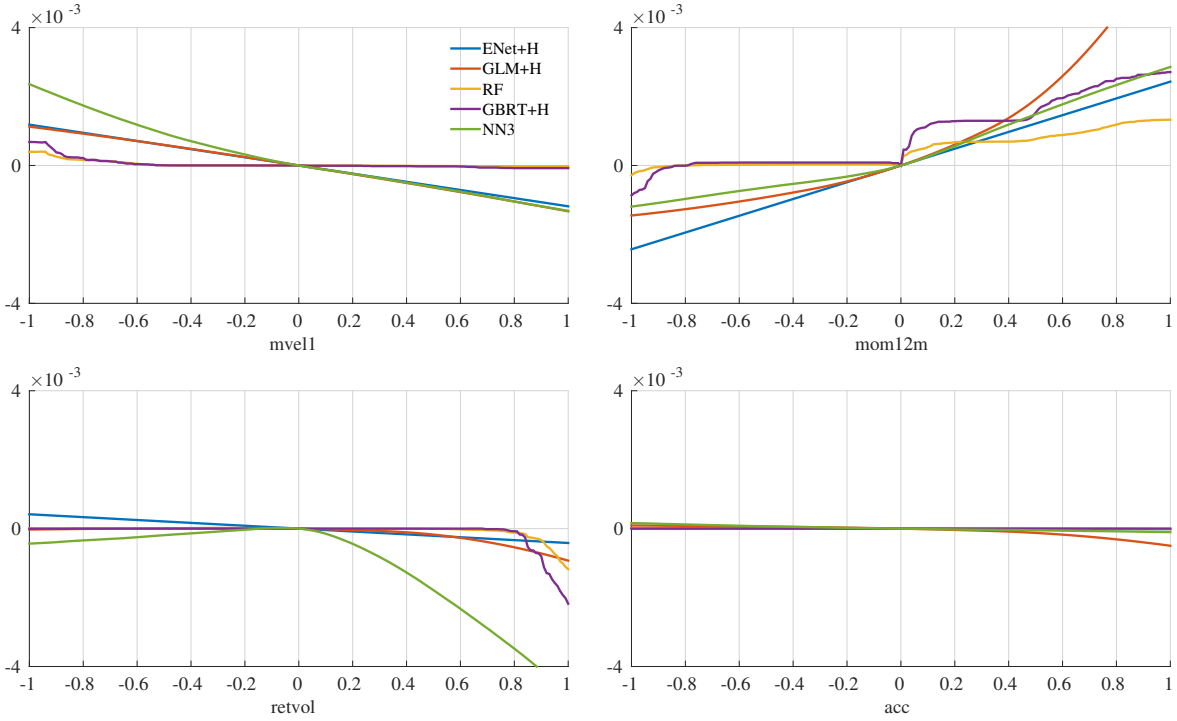
First, Figure 1.6 illustrates that machine learning methods identify patterns similar to some well-known empirical phenomena. For example, expected stock returns are decreasing in size, increasing in past 1-year return, and decreasing in stock volatility. And, interestingly, all methods agree on a nearly 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.

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.

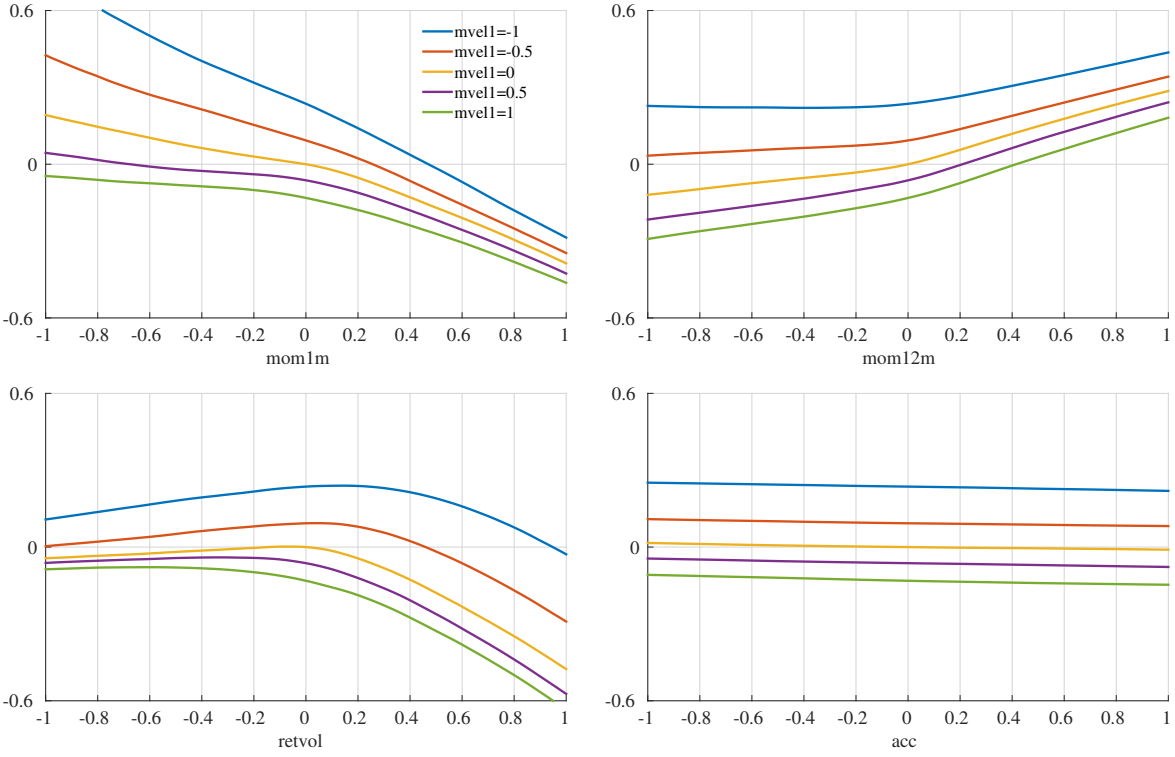
As a first example, we examine a set of pairwise interaction effects in NN3. Figure 1.7 reports how expected returns vary as we simultaneously vary values of a pair of characteristics over their support [-1,1], while holding all other variables fixed at their median value of zero. We show interactions of stock size (mvel1) with four other predictors: short-term reversal

Figure 1.6: 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).

Figure 1.7: 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).

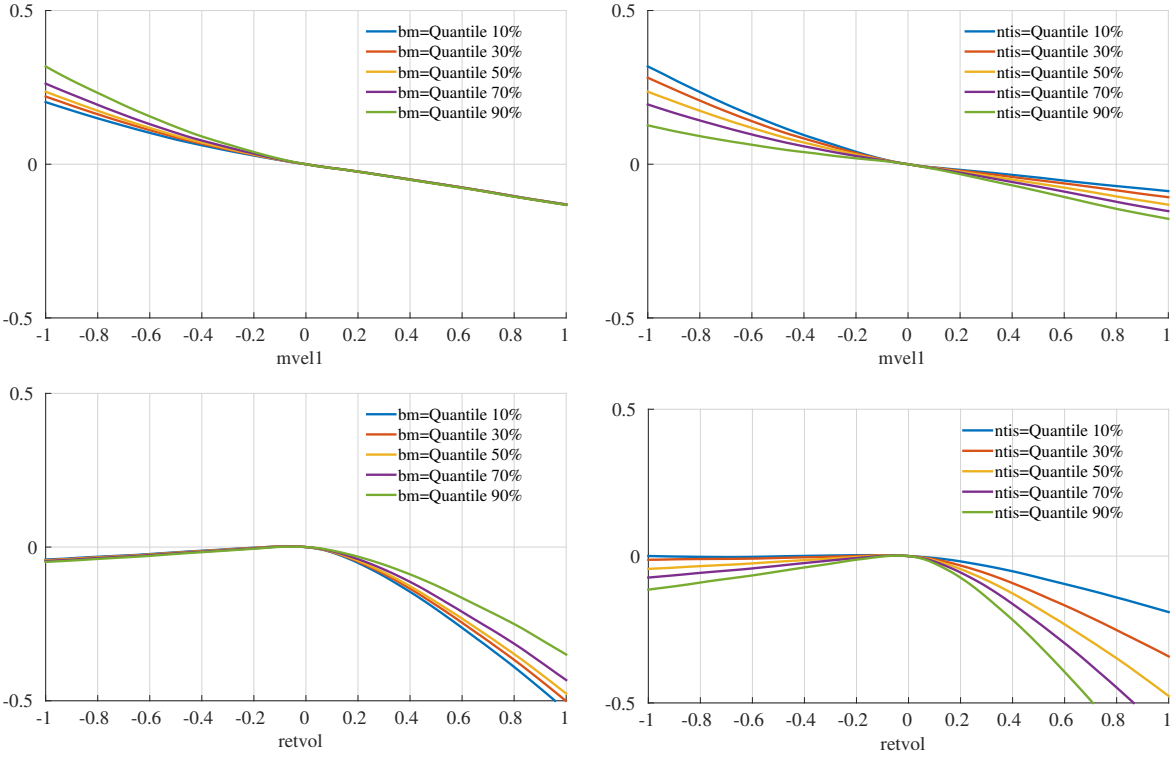
(mom1m), momentum (mom12m), and total and idiosyncratic volatility (retvol and idiovol, respectively).

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 1.8 illustrates interactions between stock-level characteristics and macroeconomic indicator variables. It shows, for example, that the size effect is more pronounced when

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 1.8: 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).

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 1.13 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 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 1.14 in the Internet Appendix.

1.3.4 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

Table 1.5: 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
S&P 500 (NEW)	-0.22	-0.85	-1.49	0.77	0.75	1.36	1.39	1.11	1.15	1.81	1.70	1.23
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.

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 perfor-

mance 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.

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]

38. Our replication of S&P 500 returns has a correlation with the actual index of more than 0.99. For the

Table 1.6: 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 [2007]).

Table 1.5 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

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.

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 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 1.5 (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 1.15 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 1.6 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 1.16 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%.

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 1.7 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, 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 1.17 in the Internet Appendix reports the performance of machine learning portfolios using an equal-weight formation. The qualitative

Table 1.7: 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. All portfolios are value weighted.

Table 1.8: 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.

conclusions of this table are identical to those of Table 1.7, 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 1.18 reports equal-weight portfolio 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 1.19 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 1.8 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.

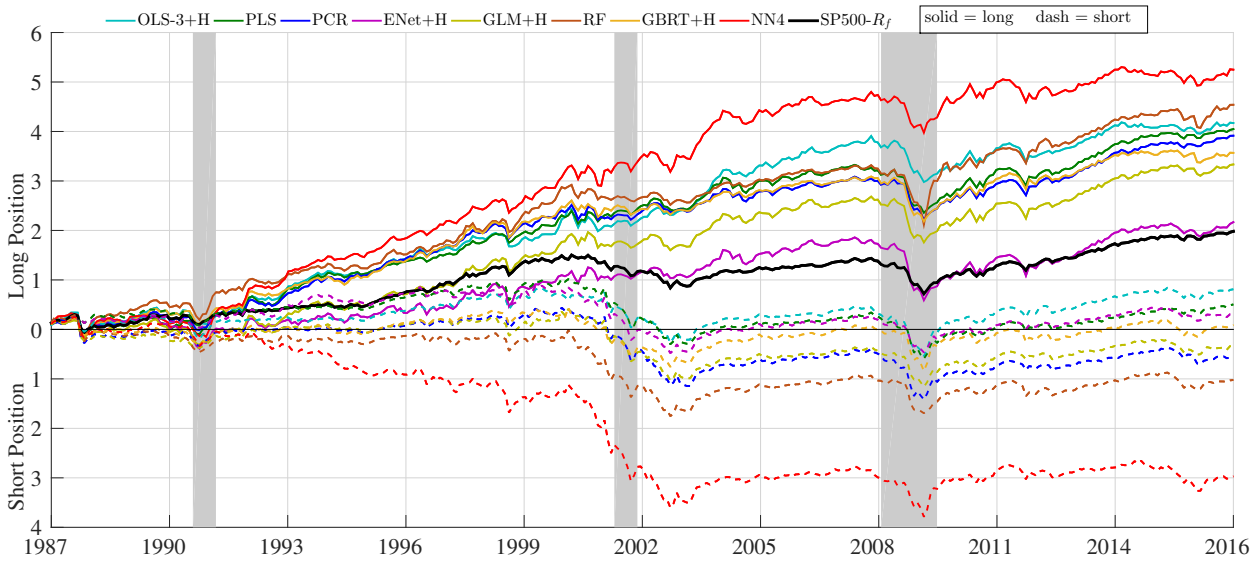
We define the strategy's average monthly turnover as

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

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 1.8 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 et al., 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

Figure 1.9: 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.

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 1.8 are visually summarized in Figure 1.9, 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 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

39. Figure 1.16 in the Internet Appendix plots the corresponding cumulative returns for equally weighted strategies.

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.

1.4 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.

1.5 Internet Appendix

1.5.1 Monte Carlo Simulations

To demonstrate the finite sample performance of all machine learning procedures, we simulate a (latent) 3–factor model for excess returns r_{t+1}, for $t = 1, 2, \ldots, T$:

$$r_{i,t+1} = g^\star(z_{i,t}) + e_{i,t+1}, \quad e_{i,t+1} = \beta_{i,t} v_{t+1} + \varepsilon_{i,t+1}, \quad z_{i,t} = (1, x_t)' \otimes c_{i,t}, \quad \beta_{i,t} = (c_{i1,t}, c_{i2,t}, c_{i3,t}),$$

where c_t is an $N \times P_c$ matrix of characteristics, v_{t+1} is a 3×1 vector of factors, x_t is a univariate time series, and ε_{t+1} is a $N \times 1$ vector of idiosyncratic errors. We choose $v_{t+1} \sim \mathcal{N}(0, 0.05^2 \times \mathbb{I}_3)$, and $\varepsilon_{i,t+1} \sim t_5(0, 0.05^2)$, in which their variances are calibrated so that the average time series R^2 is 40% and the average annualized volatility is 30%.

We simulate the panel of characteristics for each $1 \leq i \leq N$ and each $1 \leq j \leq P_c$ from the following model:

$$c_{ij,t} = \frac{2}{N+1} \text{CSrank}(\bar{c}_{ij,t}) - 1, \quad \bar{c}_{ij,t} = \rho_j \bar{c}_{ij,t-1} + \epsilon_{ij,t}, \tag{1.24}$$

where $\rho_j \sim \mathcal{U}[0.9, 1]$, $\epsilon_{ij,t} \sim \mathcal{N}(0, 1 - \rho_j^2)$ and CSrank is Cross-Section rank function, so that the characteristics feature some degree of persistence over time, yet is cross-sectionally normalized to be within $[-1, 1]$. This matches our data cleaning procedure in the empirical study.

In addition, we simulate the time series x_t from the following model:

$$x_t = \rho x_{t-1} + u_t, \tag{1.25}$$

where $u_t \sim \mathcal{N}(0, 1 - \rho^2)$, and $\rho = 0.95$ so that x_t is highly persistent.

We consider two cases of $g^\star(\cdot)$ functions:

$$\begin{aligned}
&(a) \quad g^\star(z_{i,t}) = (c_{i1,t}, c_{i2,t}, c_{i3,t} \times x_t)\theta_0, \quad \text{where} \quad \theta_0 = (0.02, 0.02, 0.02)'; \\
&(b) \quad g^\star(z_{i,t}) = \left(c_{i1,t}^2, c_{i1,t} \times c_{i2,t}, \text{sgn}(c_{i3,t} \times x_t)\right)\theta_0, \quad \text{where} \quad \theta_0 = (0.04, 0.03, 0.012)'.
\end{aligned}$$

In both cases, $g^\star(\cdot)$ only depends on 3 covariates, so there are only 3 non-zero entries in θ, denoted as θ_0. Case (a) is simple and sparse linear model. Case (b) involves a nonlinear covariate $c_{i1,t}^2$, a nonlinear and interaction term $c_{i1,t} \times c_{i2,t}$, and a discrete variable $\text{sgn}(c_{i3,t} \times x_t)$. We calibrate the values of θ_0 such that the cross-sectional R^2 is 50%, and the predictive R^2 is 5%.

Table 1.9: Comparison of Predictive R^2s for Machine Learning Algorithms in Simulations

Model	(a)				(b)			
Parameter	$P_c = 50$		$P_c = 100$		$P_c = 50$		$P_c = 100$	
R^2(%)	IS	OOS	IS	OOS	IS	OOS	IS	OOS
OLS	7.50	1.14	8.19	-1.35	3.44	-4.72	4.39	-7.75
OLS+H	7.48	1.25	8.16	-1.15	3.43	-4.60	4.36	-7.54
PCR	2.69	0.90	1.70	0.43	0.65	0.02	0.41	-0.01
PLS	6.24	3.48	6.19	2.82	1.02	-0.08	0.99	-0.17
Lasso	6.04	4.26	6.08	4.25	1.36	0.58	1.36	0.61
Lasso+H	6.00	4.26	6.03	4.25	1.32	0.59	1.31	0.61
Ridge	6.46	3.89	6.67	3.39	1.66	0.34	1.76	0.23
Ridge+H	6.42	3.91	6.61	3.42	1.63	0.35	1.73	0.25
ENet	6.04	4.26	6.08	4.25	1.35	0.58	1.35	0.61
ENet+H	6.00	4.26	6.03	4.25	1.32	0.59	1.31	0.61
GLM	5.91	4.11	5.94	4.08	3.38	1.22	3.31	1.17
GLM+H	5.85	4.12	5.88	4.09	3.32	1.24	3.24	1.20
RF	8.34	3.35	8.23	3.30	8.05	3.07	8.22	3.02
GBRT	7.08	3.35	7.02	3.33	6.51	2.76	6.42	2.84
GBRT+H	7.16	3.45	7.11	3.37	6.47	3.12	6.37	3.22
NN1	6.53	4.37	6.72	4.28	5.61	2.78	5.80	2.59
NN2	6.55	4.42	6.72	4.26	6.22	3.13	6.33	2.91
NN3	6.47	4.34	6.67	4.27	6.03	2.96	6.09	2.68
NN4	6.47	4.31	6.66	4.24	5.94	2.81	6.04	2.51
NN5	6.41	4.27	6.55	4.14	5.81	2.72	5.70	2.20
Oracle	6.22	5.52	6.22	5.52	5.86	5.40	5.86	5.40

Note: In this table, we report the average in-sample (IS) and out-of-sample (OOS) R^2 for models (a) and (b) using Ridge, Lasso, 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), respectively. "+H" indicates the use of Huber loss instead of the l_2 loss. "Oracle" stands for using the true covariates in a pooled-OLS regression. We fix $N = 200$, $T = 180$, and $P_x = 2$, comparing $P_c = 100$ with $P_c = 50$. The number of Monte Carlo repetitions is 100.

Throughout, we fix $N = 200$, $T = 180$, and $P_x = 2$, while comparing the cases of $P_c = 100$ and $P_c = 50$, corresponding to $P = 200$ and 100, respectively, to demonstrate the effect of increasing dimensionality.

For each Monte Carlo sample, we divide the whole time series into 3 consecutive subsamples of equal length for training, validation, and testing, respectively. Specifically, we estimate each of the two models in the training sample, using PLS, PCR, Ridge, Lasso, Elastic Net (ENet), generalized linear model with group lasso (GLM), random forest (RF), gradient boosted regression trees (GBRT), and the same five architectures of neural networks (NN1,...,NN5) we adopt for the empirical work, respectively, then choose tuning parameters for each method in the validation sample, and calculate the prediction errors in the testing sample. For benchmark, we also compare the pooled OLS with all covariates and that using the oracle model.

We report the average R^2s both in-sample (IS) and out-of-sample (OOS) for each model and each method over 100 Monte Carlo repetitions in Table 1.9. Both IS and OOS R^2 are relative to the estimator based on the IS average. For model (a), Lasso, ENet and NNs deliver the best and almost identical out-of-sample R^2. This is not surprising given that the true model is sparse and linear in the input covariates. The advanced tree methods such as RF and GBRT tend to overfit, so their performance is slightly worse. By contrast, for model (b), these methods clearly dominate Lasso and ENet, because the latter cannot capture the nonlinearity in the model. GLM is slightly better, but is dominated by NNs, RF, and GBRT. OLS is the worst in all settings, not surprisingly. PLS outperforms PCR in the linear model (a), but is dominated in the nonlinear case. When P_c increases, the IS R^2 tends to increase whereas the out-of-sample R^2 decreases. Hence, the performance of all methods deteriorates as overfitting exacerbates. Using Huber loss improves the out-of-sample performance for almost all methods. RF, GBRT plus Huber loss remain the best choices for the nonlinear model. The comparison among NNs demonstrates a stark trade-off between model flexibility and implementation difficulty. Deeper models potentially allow for more parsimonious representation of the data, but their objective functions are more involved to optimize. For instance, the APG algorithm used in Elastic Net is not feasible for NNs, because its loss (as a function of weight parameters) is non-convex. As shown in the table, shallower NNs tend to outperform.

Table 1.10 presents the same IS and OOS R^2s for prediction conducted for different horizons, e.g., quarterly, half-yearly, and annually. We observe the usual increasing/hump-shape patterns of R^2s against prediction horizons documented in the literature, which is driven by the persistence of covariates. The relative performance across different models maintains the same.

Next, we report the average variable selection frequencies of 6 particular covariates and the average of the remaining $P-6$ covariates for models (a) and (b) in Table 1.11, using Lasso, Elastic Net, and Group Lasso and their robust versions. We focus on these methods because they all impose the l_1 penalty and hence encourage variable selection. As expected, for model (a), the true covariates ($c_{i1,t}$, $c_{i2,t}$, $c_{i3,t} \times x_t$) are selected in over 85% of the sample paths, whereas correlated yet redundant covariates ($c_{i3,t}$, $c_{i1,t} \times x_t$, $c_{i2,t} \times x_t$) are also selected in around 60% of the samples. By contrast, the remaining covariates are rarely selected. Although model selection mistakes are unavoidable, perhaps due to the tension between variable selection and prediction or for finite sample issues, the true covariates are part of the selected models with high probabilities. For model (b), while no covariates are part of the true model, the 6 covariates we present are more relevant, and hence selected substantially more frequently than the remaining $P-6$ ones.

Table 1.10: Comparison of Predictive R^2s for Alternative Prediction Horizons in Simulations

Model	(a)						(b)					
Horizon	Quarter		Halfyear		Annual		Quarter		Halfyear		Annual	
R^2(%)	IS	OOS	IS	OOS	IS	OOS	IS	OOS	IS	OOS	IS	OOS
OLS	18.84	-0.90	27.67	0.19	35.40	-0.15	10.03	-16.47	15.02	-23.48	20.19	-30.73
OLS+H	18.82	-0.76	27.66	0.31	35.38	-0.09	10.00	-16.27	14.99	-23.32	20.15	-30.66
PCR	3.86	0.91	5.50	1.32	7.58	1.39	0.90	-0.04	1.30	-0.04	1.71	-0.24
PLS	15.12	6.56	21.52	8.40	26.46	8.14	1.91	-0.42	1.73	-0.33	2.78	-0.80
Lasso	14.10	10.33	20.42	14.68	25.06	16.76	3.10	1.17	4.03	1.12	4.87	0.40
Lasso+H	14.01	10.32	20.30	14.67	24.85	16.74	3.03	1.19	3.91	1.15	4.66	0.48
Ridge	15.76	7.81	23.26	10.67	29.27	11.65	4.07	0.43	5.66	0.44	6.72	0.00
Ridge+H	15.68	7.84	23.15	10.69	29.13	11.65	4.00	0.45	5.56	0.46	6.56	0.04
ENet	14.08	10.33	20.49	14.69	25.06	16.69	3.10	1.15	4.07	1.14	4.80	0.41
ENet+H	13.99	10.32	20.37	14.68	24.85	16.67	3.02	1.17	3.95	1.18	4.60	0.48
GLM	13.90	9.40	21.02	13.53	27.15	15.36	7.61	2.46	10.79	2.88	13.07	1.63
GLM+H	13.79	9.42	20.88	13.56	26.99	15.40	7.48	2.51	10.59	2.93	12.77	1.71
RF	17.56	8.11	25.24	11.86	31.04	14.32	15.52	5.91	20.53	7.11	22.48	6.05
GBRT	15.98	8.94	22.68	13.27	28.68	15.06	12.39	5.87	15.85	6.90	18.08	5.99
GBRT+H	15.70	8.78	22.84	13.45	29.07	15.29	12.12	5.87	16.00	7.13	18.20	6.17
NN1	15.68	9.99	23.04	14.07	29.62	15.58	13.25	5.36	17.95	6.29	20.68	5.32
NN2	15.56	9.96	22.72	14.00	28.90	16.01	13.29	5.76	17.95	6.78	20.10	5.43
NN3	15.45	9.98	22.66	13.94	28.59	16.10	13.11	5.57	17.50	6.63	20.31	5.27
NN4	15.49	9.91	22.32	14.06	28.59	15.97	13.20	5.56	17.90	6.52	19.67	5.20
NN5	15.19	9.82	22.14	13.85	28.22	15.92	13.00	5.24	17.15	6.19	18.86	5.08
Oracle	14.37	12.72	20.73	18.15	25.42	21.56	10.91	10.28	13.61	12.75	13.04	11.52

Note: In this table, we report the average in-sample (IS) and out-of-sample (OOS) R^2s for models (a) and (b) using Ridge, Lasso, 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), respectively. "+H" indicates the use of Huber loss instead of the l_2 loss. "Oracle" stands for using the true covariates in a pooled-OLS regression. We fix $N = 200$, $T = 180$, $P_x = 2$ and $P_c = 100$, comparing the performance of different horizons. The number of Monte Carlo repetitions is 100.

Finally, we report the average VIPs of the 6 particular covariates and the average of the remaining $P - 6$ covariates for models (a) and (b) in Table 1.12, using random forest (RF) and gradient boosted regression trees (GBRT), along with neural networks. We find similar results for both models (a) and (b) that the 6 covariates we present are substantially more important than the remaining $P - 6$ ones. All methods work equally well.

Overall, the simulation results suggest that the machine learning methods are successful in singling out informative variables, even though highly correlated covariates are difficult to distinguish. This is not surprising, as these methods are implemented to improve prediction, for which purpose the best model often does not agree with the true model, in particular when covariates are highly correlated.

Table 1.11: Comparison of Average Variable Selection Frequencies in Simulations

Model (a)

Parameter	Method	$c_{i1,t}$	$c_{i2,t}$	$c_{i3,t}$	$c_{i1,t} \times x_t$	$c_{i2,t} \times x_t$	$c_{i3,t} \times x_t$	Noise
$P_c = 50$	Lasso	0.95	0.94	0.65	0.53	0.51	0.85	0.09
	Lasso+H	0.95	0.94	0.63	0.53	0.50	0.86	0.08
	ENet	0.95	0.94	0.65	0.54	0.51	0.86	0.09
	ENet+H	0.95	0.94	0.64	0.53	0.50	0.86	0.09
	GLM	0.95	0.95	0.72	0.61	0.63	0.90	0.13
	GLM+H	0.95	0.94	0.70	0.61	0.62	0.90	0.12
$P_c = 100$	Lasso	0.95	0.94	0.65	0.52	0.49	0.85	0.06
	Lasso+H	0.95	0.94	0.63	0.53	0.49	0.86	0.06
	ENet	0.95	0.94	0.65	0.53	0.49	0.86	0.06
	ENet+H	0.95	0.94	0.64	0.53	0.49	0.86	0.06
	GLM	0.95	0.94	0.72	0.58	0.61	0.90	0.09
	GLM+H	0.95	0.94	0.69	0.55	0.60	0.90	0.09

Model (b)

Parameter	Method	$c_{i1,t}$	$c_{i2,t}$	$c_{i3,t}$	$c_{i1,t} \times x_t$	$c_{i2,t} \times x_t$	$c_{i3,t} \times x_t$	Noise
$P_c = 50$	Lasso	0.26	0.26	0.39	0.27	0.31	0.75	0.04
	Lasso+H	0.25	0.25	0.38	0.28	0.31	0.75	0.04
	ENet	0.26	0.25	0.39	0.27	0.31	0.76	0.04
	ENet+H	0.25	0.24	0.39	0.28	0.31	0.75	0.04
	GLM	0.80	0.54	0.68	0.68	0.64	0.82	0.21
	GLM+H	0.79	0.54	0.70	0.68	0.62	0.82	0.20
$P_c = 100$	Lasso	0.25	0.25	0.37	0.25	0.31	0.75	0.02
	Lasso+H	0.24	0.24	0.36	0.26	0.31	0.75	0.02
	ENet	0.25	0.25	0.37	0.25	0.31	0.76	0.02
	ENet+H	0.24	0.24	0.37	0.26	0.31	0.75	0.02
	GLM	0.80	0.52	0.67	0.65	0.57	0.81	0.14
	GLM+H	0.79	0.48	0.67	0.66	0.56	0.81	0.13

Note: In this table, we report the average variable selection frequencies of 6 particular covariates for models (a) and (b) (monthly horizon) using Lasso, Elastic Net (ENet), and generalized linear model with group lasso (GLM), respectively. "+H" indicates the use of Huber loss instead of the l_2 loss. Column "Noise" reports the average selection frequency of the remaining $P - 6$ covariates. We fix $N = 200$, $T = 180$, and $P_x = 2$, comparing $P_c = 100$ with $P_c = 50$. The number of Monte Carlo repetitions is 100.

Table 1.12: Comparison of Average Variable Importance in Simulations

Model (a)

Parameter	Method	$c_{i1,t}$	$c_{i2,t}$	$c_{i3,t}$	$c_{i1,t} \times x_t$	$c_{i2,t} \times x_t$	$c_{i3,t} \times x_t$	Noise
$P_c = 50$	RF	21.54	23.20	5.89	6.44	6.42	19.45	0.18
	GBRT	23.86	27.86	5.64	6.41	6.03	25.66	0.05
	GBRT+H	24.01	27.43	5.33	6.30	6.78	25.88	0.05
	NN1	26.50	29.55	5.31	2.99	3.75	25.18	0.07
	NN2	26.35	28.93	5.04	3.12	3.93	25.74	0.07
	NN3	26.05	28.61	5.03	3.09	3.89	25.57	0.08
	NN4	26.09	28.72	5.08	3.37	3.82	25.59	0.08
	NN5	25.95	28.40	5.12	3.31	3.73	25.36	0.09
$P_c = 100$	RF	21.40	23.08	5.84	5.62	5.87	19.18	0.10
	GBRT	23.87	27.82	5.32	6.20	5.87	25.43	0.03
	GBRT+H	23.75	27.12	5.20	6.04	6.30	26.00	0.03
	NN1	25.78	28.36	5.03	2.77	3.60	24.57	0.05
	NN2	25.30	27.88	4.85	2.95	3.52	24.58	0.06
	NN3	25.32	28.03	4.73	2.89	3.50	24.53	0.06
	NN4	25.02	27.63	4.77	2.92	3.49	24.34	0.06
	NN5	24.78	27.73	4.82	3.07	3.54	24.19	0.06

Model (b)

Parameter	Method	$c_{i1,t}$	$c_{i2,t}$	$c_{i3,t}$	$c_{i1,t} \times x_t$	$c_{i2,t} \times x_t$	$c_{i3,t} \times x_t$	Noise
$P_c = 50$	RF	27.70	6.47	5.03	8.02	5.00	32.13	0.17
	GBRT	31.24	7.37	5.82	8.81	6.43	36.41	0.04
	GBRT+H	32.05	7.46	5.83	8.87	6.62	35.57	0.04
	NN1	55.55	14.50	4.53	3.46	2.97	12.11	0.07
	NN2	51.84	13.66	4.15	2.96	2.72	18.32	0.07
	NN3	52.00	13.64	4.36	2.93	2.89	16.63	0.08
	NN4	51.07	13.61	4.45	3.31	2.81	16.19	0.09
	NN5	49.74	13.68	4.48	3.28	2.86	15.91	0.11
$P_c = 100$	RF	26.42	5.74	4.40	7.77	4.69	31.93	0.10
	GBRT	31.49	7.30	5.38	8.61	6.17	36.70	0.02
	GBRT+H	32.13	7.48	5.71	8.66	6.30	35.87	0.02
	NN1	53.09	13.53	4.79	3.45	2.77	12.11	0.05
	NN2	50.25	12.93	4.26	2.87	2.45	17.31	0.05
	NN3	50.36	13.00	4.30	2.90	2.54	15.43	0.06
	NN4	48.28	13.05	4.50	3.21	2.63	15.05	0.07
	NN5	43.44	12.41	4.71	3.61	2.59	16.09	0.09

Note: In this table, we report the average variable importance of 6 particular covariates for models (a) and (b) (monthly horizon) using random forest (RF), gradient boosted regression trees (GBRT), and five architectures of neural networks (NN1,...,NN5), respectively. "+H" indicates the use of Huber loss instead of the l_2 loss. Column "Noise" reports the average variable importance of the remaining $P - 6$ covariates. We fix $N = 200$, $T = 180$, and $P_x = 2$, comparing $P_c = 100$ with $P_c = 50$. The number of Monte Carlo repetitions is 100.

1.5.2 Algorithms in Details

Lasso, Ridge, Elastic Net, and Group Lasso

We present the accelerated proximal algorithm (APG), see, e.g., Parikh and Boyd [2013] and Polson et al. [2015]., which allows for efficient implementation of the elastic net, Lasso, Ridge regression, and Group Lasso for both l_2 and Huber losses. We rewrite their regularized objective functions as

$$\mathcal{L}(\theta;\cdot) = \underbrace{\mathcal{L}(\theta)}_{\text{Loss Function}} + \underbrace{\phi(\theta;\cdot)}_{\text{Penalty}}, \tag{1.26}$$

where we omit the dependence on the tuning parameters. Specifically, we have

$$\phi(\theta;\cdot) = \begin{cases} \frac{1}{2}\lambda\sum_{j=1}^{P}\theta_j^2, & \text{Ridge;} \\ \lambda\sum_{j=1}^{P}|\theta_j|, & \text{Lasso;} \\ \lambda(1-\rho)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2, & \text{Elastic Net;} \\ \lambda\sum_{j=1}^{P}\|\theta_j\|, & \text{Group Lasso.} \end{cases}, \tag{1.27}$$

where in the Group Lasso case, $\theta = (\theta_1, \theta_2, \ldots, \theta_P)$ is a $K \times P$ matrix.

Proximal algorithms are a class of algorithms for solving convex optimization problems, in which the base operation is evaluating the proximal operator of a function, ie., solving a small convex optimization problem. In many cases, this smaller problem has a closed form solution. The proximal operator is defined as:

$$\mathbf{prox}_{\gamma f}(\theta) = \underset{z}{\operatorname{argmin}} \left\{ f(z) + \frac{1}{2\gamma}\|z-\theta\|^2 \right\}.$$

An important property of the proximal operator is that the minimizer of a convex function $f(\cdot)$ is a fixed point of $\mathbf{prox}_f(\cdot)$, i.e., $\theta^\star$ minimizes $f(\cdot)$ if and only if

$$\theta^\star = \mathbf{prox}_f(\theta^\star).$$

The proximal gradient algorithm is designed to minimize an objective function of the form

(1.26), where $\mathcal{L}(\theta)$ is differentiable function of θ but $\phi(\theta;\cdot)$ is not. Using properties of the proximal operator, one can show that $\theta^\star$ minimizes (1.26), if and only if

$$\theta^\star = \mathbf{prox}_{\gamma\phi}(\theta^\star - \gamma\nabla\mathcal{L}(\theta^\star)).$$

This result motivates the first two iteration steps in Algorithm 1. The third step inside the while loop is a Nesterov momentum (Nesterov [1983]) adjustment that accelerates convergence.

The optimization problem requires the proximal operators of $\phi(\theta;\cdot)$s in (1.27), which have closed forms:

$$\mathbf{prox}_{\gamma\phi}(\theta) = \begin{cases} \dfrac{\theta}{1+\lambda\gamma}, & \text{Ridge;} \\ \lambda S(\theta, \lambda\gamma), & \text{Lasso;} \\ \dfrac{1}{1+\lambda\gamma\rho} S(\theta, (1-\rho)\lambda\gamma), & \text{Elastic Net;} \\ (\widetilde{S}(\theta_1, \lambda\gamma)^\intercal, \widetilde{S}(\theta_2, \lambda\gamma)^\intercal, \ldots, \widetilde{S}(\theta_P, \lambda\gamma)^\intercal)^\intercal, & \text{Group Lasso.} \end{cases},$$

where $S(x,\mu)$ and $\widetilde{S}(x,\mu)$ are vector-valued functions, whose ith components are defined by:

$$(S(x,\mu))_i = \begin{cases} x_i - \mu, & \text{if } x_i > 0 \text{ and } \mu < |x_i|; \\ x_i + \mu, & \text{if } x_i < 0 \text{ and } \mu < |x_i|; \\ 0, & \text{if } \mu \geq |x_i|. \end{cases}, \quad (\widetilde{S}(x,\mu))_i = \begin{cases} x_i - \mu\frac{x_i}{\|x_i\|}, & \text{if } \|x_i\| > \mu; \\ 0, & \text{if } \|x_i\| \leq \mu. \end{cases}.$$

Note that $S(x,\mu)$ is the soft-thresholding operator, so the proximal algorithm is equivalent to the coordinate descent algorithm in the case of l_2 loss, see, e.g., Daubechies et al. [2004], Friedman et al. [2007]. The proximal framework we adopt here allows efficient implementation of Huber loss and convergence acceleration.

Algorithm 1: Accelerated Proximal Gradient Method

Initialization: $\theta_0 = 0$, $m = 0$, γ;
while θ_m *not converged* **do**
  $\bar{\theta} \leftarrow \theta_m - \gamma\nabla\mathcal{L}(\theta)\,|_{\theta=\theta_m}$.
  $\tilde{\theta} \leftarrow \mathbf{prox}_{\gamma\phi}(\bar{\theta})$.
  $\theta_{m+1} \leftarrow \tilde{\theta} + \frac{m}{m+3}(\tilde{\theta} - \theta_m)$.
  $m \leftarrow m + 1$.
end
Result: The final parameter estimate is θ_m.

Tree, Random Forest, and Gradient Boosted Tree

Algorithm 2 is a greedy algorithm, see, e.g., Breiman et al. [1984], to grow a complete binary regression tree. Next, Algorithm 3 yields the random forest, e.g., Hastie et al. [2009]. Finally, Algorithm 4 delivers the gradient boosted tree (Friedman [2001]), for which we follow the version written by Bühlmann and Hothorn [2007].

Algorithm 2: Classification and Regression Tree

Initialize the stump. $C_1(0)$ denotes the range of all covariates, $C_l(d)$ denote the l-th node of depth d.

for d *from* 1 *to* L **do**

  for i *in* $\{C_l(d-1), l = 1, ..., 2^{d-1}\}$ **do**

    i) For each feature $j = 1, 2, \ldots, P$, and each threshold level α, define a split as $s = (j, \alpha)$, which divides $C_l(d-1)$ into C_{left} and C_{right}:

$$C_{left}(s) = \{z_j \leq \alpha\} \cap C_l(d-1); \quad C_{right}(s) = \{z_j > \alpha\} \cap C_l(d-1),$$

    where z_j denotes the jth covariate.

    ii) Define the impurity function:

$$\begin{aligned}\mathcal{L}(C, C_{left}, C_{right}) =& \frac{|C_{left}|}{|C|} H(C_{left}) + \frac{|C_{right}|}{|C|} H(C_{right}), \text{ where} \\ H(C) =& \frac{1}{|C|} \sum_{z_{i,t} \in C} (r_{i,t+1} - \theta)^2, \quad \theta = \frac{1}{|C|} \sum_{z_{i,t} \in C} r_{i,t+1},\end{aligned}$$

    and $|C|$ denotes the number of observations in set C.

    iii) Select the optimal split:

$$s^* \leftarrow \underset{s}{\operatorname{argmin}}\ \mathcal{L}(C(s), C_{left}(s), C_{right}(s)).$$

    iv) Update the nodes:

$$C_{2l-1}(d) \leftarrow C_{left}(s^*), \quad C_{2l}(d) \leftarrow C_{right}(s^*).$$

  end

end

Result: The output of a regression tree is given by:

$$g(z_{i,t}; \theta, L) = \sum_{k=1}^{2^L} \theta_k \mathbf{1}\left\{z_{i,t} \in C_k(L)\right\}, \text{ where } \theta_k = \frac{1}{|C_k(L)|} \sum_{z_{i,t} \in C_k(L)} r_{i,t+1}.$$

For a single binary complete regression tree $\mathcal{T}$ of depth L, the VIP for the covariate z_j is

$$\text{VIP}(z_j, \mathcal{T}) = \sum_{d=1}^{L-1} \sum_{i=1}^{2^{d-1}} \Delta im\left(C_i(d-1), C_{2i-1}(d), C_{2i}(d)\right) \mathbf{1}\{z_j \in \mathcal{T}(i, d)\},$$

where $\mathcal{T}(i, d)$ represents the covariate on the i-th (internal) node of depth d, which splits $C_i(d-1)$ into two sub-regions $\{C_{2i-1}(d), C_{2i}(d)\}$, and $\Delta im(\cdot, \cdot, \cdot)$ is defined by:

$$\Delta im\left(C, C_{left}, C_{right}\right) = H(C) - \mathcal{L}(C, C_{left}, C_{right}).$$

Algorithm 3: Random Forest

for b *from* 1 *to* B **do**

Generate Bootstrap samples $\{(z_{i,t}, r_{i,t+1}), (i,t) \in Bootstrap(b)\}$ from the original dataset, for which a tree is grown using Algorithm 2. At each step of splitting, use only a random subsample, say $\sqrt{P}$ or any specific number, of all features. Write the resulting bth tree as:

$$\widehat{g}_b(z_{i,t}; \widehat{\theta}_b, L) = \sum_{k=1}^{2^L} \theta_b^{(k)} \mathbf{1}\left\{z_{i,t} \in C_k(L)\right\}.$$

end

Result: The final random forest output is given by the average of the outputs of all B trees.

$$\widehat{g}(z_{i,t}; L, B) = \frac{1}{B} \sum_{b=1}^{B} \widehat{g}_b(z_{i,t}; \widehat{\theta}_b, L).$$

Algorithm 4: Gradient Boosted Tree

Initialize the predictor as $\widehat{g}_0(\cdot) = 0$;

for b *from* 1 *to* B **do**

Compute for each $i = 1, 2, \ldots, N$ and $t = 1, 2, \ldots, T$, the negative gradient of the loss function $l(\cdot, \cdot)$:[a]

$$\varepsilon_{i,t+1} \leftarrow -\frac{\partial l(r_{i,t+1}, g)}{\partial g} \Big|_{g=\widehat{g}_{b-1}(z_{i,t})}.$$

Grow a (shallow) regression tree of depth L with dataset $\{(z_{i,t}, \varepsilon_{i,t+1}) : \forall i, \ \forall t\}$

$$\widehat{f}_b(\cdot) \leftarrow g(z_{i,t}; \theta, L).$$

Update the model by

$$\widehat{g}_b(\cdot) \leftarrow \widehat{g}_{b-1}(\cdot) + \nu \widehat{f}_b(\cdot),$$

where $\nu \in (0, 1]$ is a tuning parameter that controls the step length.

end

Result: The final model output is

$$\widehat{g}_B(z_{i,t}; B, \nu, L) = \sum_{b=1}^{B} \nu \widehat{f}_b(\cdot).$$

a. The typical choice of $l(\cdot, \cdot)$ for regression is l_2 or Huber loss, whereas for classification, it is more common to use the following loss function:

$$l(d, g(\cdot)) = \log_2 \left(1 + \exp\left(-2(2d - 1)g(\cdot)\right)\right).$$

Neural Networks

It is common to fit the neural network using stochastic gradient descent (SGD), see, e.g., Goodfellow et al. [2016]. We adopt the adaptive moment estimation algorithm (Adam), an efficient version of the SGD introduced by Kingma and Ba [2014]. Adam computes adaptive learning rates for individual parameters using estimates of first and second moments of the gradients. We denote the loss function as $\mathcal{L}(\theta;\cdot)$ and write $\mathcal{L}(\theta;\cdot) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_t(\theta;\cdot)$, where $\mathcal{L}_t(\theta;\cdot)$ is the penalized cross-sectional average prediction error at time t. At each step of training, a batch sent to the algorithm is randomly sampled from the training dataset. Algorithm 6 is the early stopping algorithm that can be used in combination with many optimization routines, including Adam. Algorithm 7 gives the Batch-Normalization transform (Ioffe and Szegedy [2015]), which we apply to each activation after ReLU transformation. Any neuron that previously receives a batch of x as the input now receives $\mathrm{BN}_{\gamma,\beta}(x)$ instead, where γ and β are additional parameters to be optimized.

Algorithm 5: Adam for Stochastic Gradient Descent (SGD)

Initialize the parameter vector θ_0. Set $m_0 = 0, v_0 = 0, t = 0$.

while θ_t *not converged* **do**

$t \leftarrow t + 1$.

$g_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta;\cdot)\big|_{\theta=\theta_{t-1}}$.

$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$.

$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t \odot g_t$.[a]

$\hat{m}_t \leftarrow m_t / (1-(\beta_1)^t)$.

$\hat{v}_t \leftarrow v_t / (1-(\beta_2)^t)$.

$\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t \oslash (\sqrt{\hat{v}_t} + \epsilon)$.

end

Result: The final parameter estimate is θ_t.

a. $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively.

Algorithm 6: Early Stopping

Initialize $j = 0$, $\epsilon = \infty$ and select the patience parameter p.

while $j < p$ **do**

  Update θ using the training algorithm (e.g., the steps inside the while loop of Algorithm 5 for h steps).

  Calculate the prediction error from the validation sample, denoted as ϵ'.

  if $\epsilon' < \epsilon$ **then**

    $j \leftarrow 0$.

    $\epsilon \leftarrow \epsilon'$.

    $\theta' \leftarrow \theta$.

    else

    $j \leftarrow j + 1$.

  end

end

Result: The final parameter estimate is θ'.

Algorithm 7: Batch Normalization (for one Activation over one Batch)

Input: Values of x for each activation over a batch $\mathcal{B} = \{x_1, x_2, \ldots, x_N\}$.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta := \mathrm{BN}_{\gamma,\beta}(x_i)$$

Result: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i) : i = 1, 2 \ldots, N\}$.

1.5.3 Theoretical Properties of Machine Learning Models

In this section, we provide references on the asymptotic properties of machine learning methods discussed in the main text. The references below are unavoidably selective and by no means complete. We invite interested readers to consult references within the following papers.

For theoretical properties of lasso, see Knight and Fu [2000], Bickel et al. [2009], Meinshausen and Yu [2009], Tibshirani [2011], Wainwright [2009], and Zhang and Huang [2008]. And for elastic net, see Zou and Hastie [2005], Zou and Zhang [2009], and Mol et al. [2009]. For group Lasso in linear models, see Lounici et al. [2011], and see Bach [2008] and Ravikumar et al. [2009] for additive and nonparametric models. While most theoretical analysis in high-dimensional statistics assume that data have sub-Gaussian or sub-exponential tails, Fan et al. [2017] provide a theoretical justification of using Huber's loss function in the high-dimensional

setting.

For dimension reduction techniques, there is a large literature in statistics on the asymptotic behavior of PCA, e.g., Bai [1999], Johnstone [2001], Johnstone and Lu [2009], Paul [2007], Wang and Fan [2017], and another large literature in econometrics focusing on the asymptotic theory of PCA in modern factor analysis, e.g., Stock and Watson [2002b], Bai and Ng [2002], and Bai and Ng [2013]. There are, however, fewer results on the asymptotic analysis of PCR and PLS in particular. One can refer to Giglio and Xiu [2016] for the asymptotic theory of PCR in the context of risk premia estimation, and Kelly and Pruitt [2013, 2015] for the theory of PLS with its application to forecasting risk premia in financial markets.

A recent literature analyzes theoretical properties of random forests, see Biau [2012], Scornet et al. [2015], Mentch and Hooker [2016], Wager et al. [2014], and Wager and Athey [2018]. The properties of gradient boosting, on the other hand, are well understood from the early work of e.g., Friedman et al. [2000b], Bühlmann and Yu [2003], Lugosi and Vayatis [2004], and Zhang and Yu [2005] for both classification and regression problems. However, much work remains to be done to fully take into account optimization and regularization algorithms that are essential to the desirable performance of various boosting methods, e.g., the popular XGBoost system designed by Chen and Guestrin [2016].

Likewise, theoretical properties of neural networks and deep learning are in large part under-developed [for an overview, see Fan et al., 2019]. First, the approximation theory of neural networks is far from complete. Although earlier work have established a universal approximation theory with a single hidden layer network [e.g., Hornik et al., 1989], a recent line of work sheds light on the distinction between depth and width of a multi-layer network. 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 [see also Lin et al., 2017, Rolnick and Tegmark, 2018].

Second, any theoretical understanding of neural networks should explicitly account for the modern optimization algorithms that, in combination with statistical analysis, are critical to their success. But training a deep neural network typically involves a grab bag of algorithms, e.g., SGD, Adam, batch normalization, skipping connections [He et al., 2016b], some of which rely on heuristic explanation without rigorous analysis. A promising recent strand of work Chizat and Bach [2018], Mei et al. [2018], and Mei et al. [2019] approximate the evolution of network weight parameters in the SDG algorithm for networks with a single hidden layer. They show that mean-field partial differential equations accurately describe this process as long as the number of hidden units is sufficiently large. In summary, there remains much work to be done to establish theoretical properties of deep learning.

Table 1.13: Hyperparameters For All Methods

	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1 - NN5
Huber loss $\xi =$ 99.9% quantile	✓	-	-	✓	✓	-	✓	-
Others		K	K	$\rho = 0.5$ $\lambda \in (10^{-4}, 10^{-1})$	#Knots=3 $\lambda \in (10^{-4}, 10^{-1})$	Depth= $1 \sim 6$ #Trees= 300 #Features in each split $\in \{3, 5, 10, 20, 30, 50...\}$	Depth= $1 \sim 2$ #Trees= $1 \sim 1000$ Learning Rate LR$\in \{0.01, 0.1\}$	L1 penalty $\lambda_1 \in (10^{-5}, 10^{-3})$ Learning Rate LR$\in \{0.001, 0.01\}$ Batch Size=10000 Epochs=100 Patience=5 Adam Para.=Default Ensemble=10

Note: The table describes the hyperparameters that we tune in each machine learning method.

1.5.4 Sample Splitting

We consider a number of sample splitting schemes studied in the forecast evaluation literature [see, e.g., West, 2006]. The "fixed" scheme splits the data into training, validation, and testing samples. It estimates the model once from the training and validation samples, and attempts to fit all points in the testing sample using this fixed model estimate.

A common alternative to the fixed split scheme is a "rolling" scheme, in which the training and validation samples gradually shift forward in time to include more recent data, but holds the total number of time periods in each training and validation sample fixed. For each rolling window, one re-fits the model from the prevailing training and validation samples, and tracks a model's performance in the remaining test data that has not been subsumed by the rolling windows. The result is a sequence of performance evaluation measures corresponding to each rolling estimation window. This has the benefit of leveraging more recent information for prediction relative to the fixed scheme.

The third is a "recursive" performance evaluation scheme. Like the rolling approach, it gradually includes more recent observations in the training and validation windows. But the recursive scheme always retains the entire history in the training sample, thus its window size gradually increases. The rolling and recursive schemes are computationally expensive, in particular for more complicated models such as neural networks.

In our empirical exercise, we adopt a hybrid of these schemes by recursively increasing the training sample, periodically refitting the entire model once per year, and making out-of-sample predictions using the same fitted model over the subsequent year. Each time we refit, we increase the training sample by a year, while maintaining a fixed size rolling sample for validation. We choose to *not* cross-validate in order to maintain the temporal ordering of the

data for prediction.

1.5.5 Hyperparameter Tuning

Table 1.13 describes the set of hyperparameters and their potential values used for tuning each machine learning model.

1.5.6 Additional Tables and Figures

Table 1.14: Details of the Characteristics

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
1	absacc	Absolute accruals	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Annual
2	acc	Working capital accruals	Sloan	1996, TAR	Compustat	Annual
3	aeavol	Abnormal earnings announcement volume	Lerman, Livnat & Mendenhall	2007, WP	Compustat+CRSP	Quarterly
4	age	# years since first Compustat coverage	Jiang, Lee & Zhang	2005, RAS	Compustat	Annual
5	agr	Asset growth	Cooper, Gulen & Schill	2008, JF	Compustat	Annual
6	baspread	Bid-ask spread	Amihud & Mendelson	1989, JF	CRSP	Monthly
7	beta	Beta	Fama & MacBeth	1973, JPE	CRSP	Monthly
8	betasq	Beta squared	Fama & MacBeth	1973, JPE	CRSP	Monthly
9	bm	Book-to-market	Rosenberg, Reid & Lanstein	1985, JPM	Compustat+CRSP	Annual
10	bm_ia	Industry-adjusted book to market	Asness, Porter & Stevens	2000, WP	Compustat+CRSP	Annual
11	cash	Cash holdings	Palazzo	2012, JFE	Compustat	Quarterly
12	cashdebt	Cash flow to debt	Ou & Penman	1989, JAE	Compustat	Annual
13	cashpr	Cash productivity	Chandrashekar & Rao	2009, WP	Compustat	Annual
14	cfp	Cash flow to price ratio	Desai, Rajgopal & Venkatachalam	2004, TAR	Compustat	Annual
15	cfp_ia	Industry-adjusted cash flow to price ratio	Asness, Porter & Stevens	2000, WP	Compustat	Annual
16	chatoia	Industry-adjusted change in asset turnover	Soliman	2008, TAR	Compustat	Annual
17	chcsho	Change in shares outstanding	Pontiff & Woodgate	2008, JF	Compustat	Annual
18	chempia	Industry-adjusted change in employees	Asness, Porter & Stevens	1994, WP	Compustat	Annual
19	chinv	Change in inventory	Thomas & Zhang	2002, RAS	Compustat	Annual
20	chmom	Change in 6-month momentum	Gettleman & Marks	2006, WP	CRSP	Monthly
21	chpmia	Industry-adjusted change in profit margin	Soliman	2008, TAR	Compustat	Annual
22	chtx	Change in tax expense	Thomas & Zhang	2011, JAR	Compustat	Quarterly
23	cinvest	Corporate investment	Titman, Wei & Xie	2004, JFQA	Compustat	Quarterly
24	convind	Convertible debt indicator	Valta	2016, JFQA	Compustat	Annual
25	currat	Current ratio	Ou & Penman	1989, JAE	Compustat	Annual
26	depr	Depreciation / PP&E	Holthausen & Larcker	1992, JAE	Compustat	Annual
27	divi	Dividend initiation	Michaely, Thaler & Womack	1995, JF	Compustat	Annual
28	divo	Dividend omission	Michaely, Thaler & Womack	1995, JF	Compustat	Annual
29	dolvol	Dollar trading volume	Chordia, Subrahmanyam & Anshuman	2001, JFE	CRSP	Monthly
30	dy	Dividend to price	Litzenberger & Ramaswamy	1982, JF	Compustat	Annual
31	ear	Earnings announcement return	Kishore, Brandt, Santa-Clara & Venkatachalam	2008, WP	Compustat+CRSP	Quarterly

Note: This table lists the characteristics we use in the empirical study. The data are collected in Green et al. [2017].

Table 1.14: Details of the Characteristics (Continued)

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
32	egr	Growth in common shareholder equity	Richardson, Sloan, Soliman & Tuna	2005, JAE	Compustat	Annual
33	ep	Earnings to price	Basu	1977, JF	Compustat	Annual
34	gma	Gross profitability	Novy-Marx	2013, JFE	Compustat	Annual
35	grCAPX	Growth in capital expenditures	Anderson & Garcia-Feijoo	2006, JF	Compustat	Annual
36	grltnoa	Growth in long term net operating assets	Fairfield, Whisenant & Yohn	2003, TAR	Compustat	Annual
37	herf	Industry sales concentration	Hou & Robinson	2006, JF	Compustat	Annual
38	hire	Employee growth rate	Bazdresch, Belo & Lin	2014, JPE	Compustat	Annual
39	idiovol	Idiosyncratic return volatility	Ali, Hwang & Trombley	2003, JFE	CRSP	Monthly
40	ill	Illiquidity	Amihud	2002, JFM	CRSP	Monthly
41	indmom	Industry momentum	Moskowitz & Grinblatt	1999, JF	CRSP	Monthly
42	invest	Capital expenditures and inventory	Chen & Zhang	2010, JF	Compustat	Annual
43	lev	Leverage	Bhandari	1988, JF	Compustat	Annual
44	lgr	Growth in long-term debt	Richardson, Sloan, Soliman & Tuna	2005, JAE	Compustat	Annual
45	maxret	Maximum daily return	Bali, Cakici & Whitelaw	2011, JFE	CRSP	Monthly
46	mom12m	12-month momentum	Jegadeesh	1990, JF	CRSP	Monthly
47	mom1m	1-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
48	mom36m	36-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
49	mom6m	6-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
50	ms	Financial statement score	Mohanram	2005, RAS	Compustat	Quarterly
51	mvel1	Size	Banz	1981, JFE	CRSP	Monthly
52	mve_ia	Industry-adjusted size	Asness, Porter & Stevens	2000, WP	Compustat	Annual
53	nincr	Number of earnings increases	Barth, Elliott & Finn	1999, JAR	Compustat	Quarterly
54	operprof	Operating profitability	Fama & French	2015, JFE	Compustat	Annual
55	orgcap	Organizational capital	Eisfeldt & Papanikolaou	2013, JF	Compustat	Annual
56	pchcapx_ia	Industry adjusted % change in capital expenditures	Abarbanell & Bushee	1998, TAR	Compustat	Annual
57	pchcurrat	% change in current ratio	Ou & Penman	1989, JAE	Compustat	Annual
58	pchdepr	% change in depreciation	Holthausen & Larcker	1992, JAE	Compustat	Annual
59	pchgm_pchsale	% change in gross margin - % change in sales	Abarbanell & Bushee	1998, TAR	Compustat	Annual
60	pchquick	% change in quick ratio	Ou & Penman	1989, JAE	Compustat	Annual
61	pchsale_pchinvt	% change in sales - % change in inventory	Abarbanell & Bushee	1998, TAR	Compustat	Annual
62	pchsale_pchrect	% change in sales - % change in A/R	Abarbanell & Bushee	1998, TAR	Compustat	Annual

Table 1.14: Details of the Characteristics (Continued)

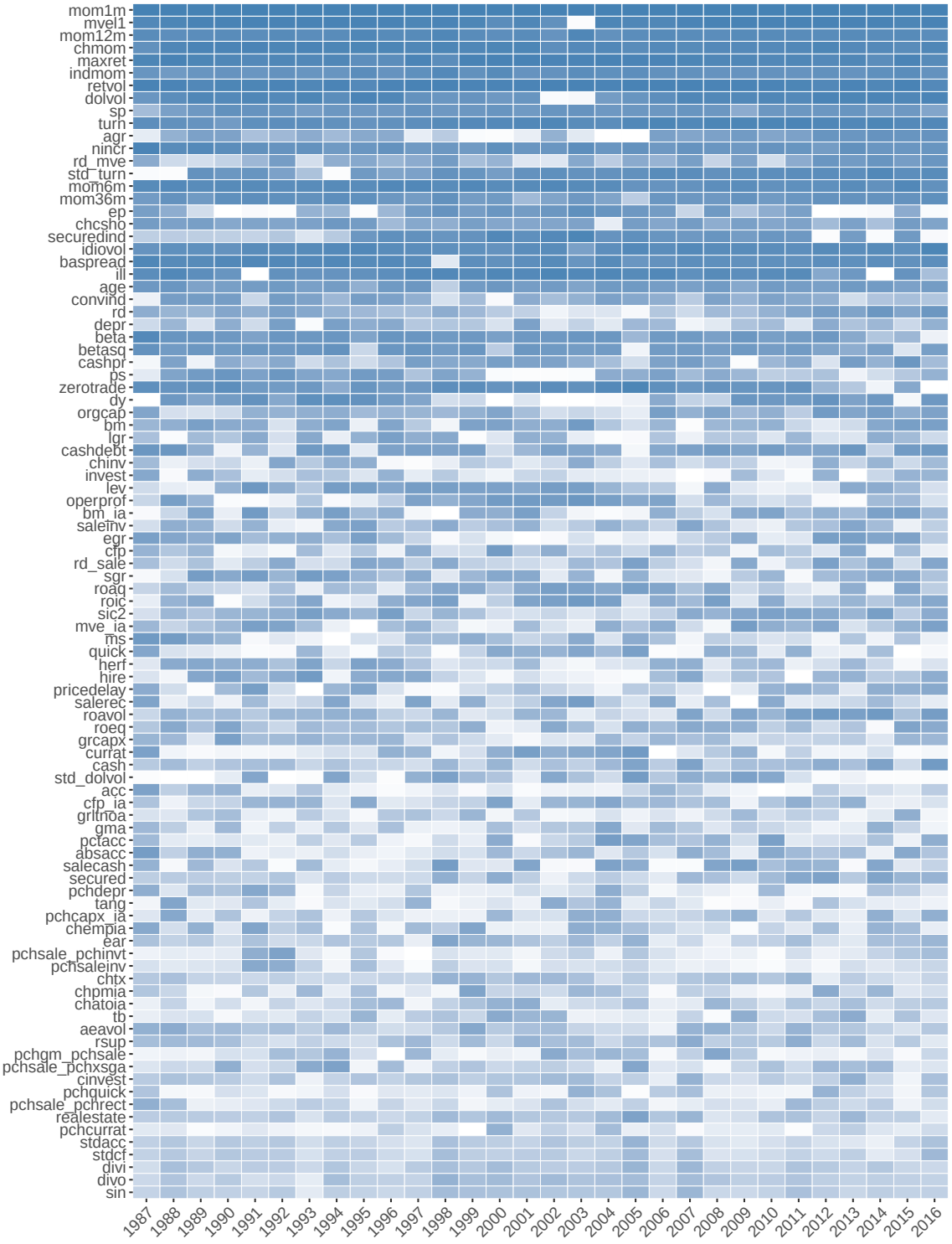
No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
63	pchsale_pchxsga	% change in sales - % change in SG&A	Abarbanell & Bushee	1998, TAR	Compustat	Annual
64	pchsaleinv	% change sales-to-inventory	Ou & Penman	1989, JAE	Compustat	Annual
65	pctacc	Percent accruals	Hafzalla, Lundholm & Van Winkle	2011, TAR	Compustat	Annual
66	pricedelay	Price delay	Hou & Moskowitz	2005, RFS	CRSP	Monthly
67	ps	Financial statements score	Piotroski	2000, JAR	Compustat	Annual
68	quick	Quick ratio	Ou & Penman	1989, JAE	Compustat	Annual
69	rd	R&D increase	Eberhart, Maxwell & Siddique	2004, JF	Compustat	Annual
70	rd_mve	R&D to market capitalization	Guo, Lev & Shi	2006, JBFA	Compustat	Annual
71	rd_sale	R&D to sales	Guo, Lev & Shi	2006, JBFA	Compustat	Annual
72	realestate	Real estate holdings	Tuzel	2010, RFS	Compustat	Annual
73	retvol	Return volatility	Ang, Hodrick, Xing & Zhang	2006, JF	CRSP	Monthly
74	roaq	Return on assets	Balakrishnan, Bartov & Faurel	2010, JAE	Compustat	Quarterly
75	roavol	Earnings volatility	Francis, LaFond, Olsson & Schipper	2004, TAR	Compustat	Quarterly
76	roeq	Return on equity	Hou, Xue & Zhang	2015, RFS	Compustat	Quarterly
77	roic	Return on invested capital	Brown & Rowe	2007, WP	Compustat	Annual
78	rsup	Revenue surprise	Kama	2009, JBFA	Compustat	Quarterly
79	salecash	Sales to cash	Ou & Penman	1989, JAE	Compustat	Annual
80	saleinv	Sales to inventory	Ou & Penman	1989, JAE	Compustat	Annual
81	salerec	Sales to receivables	Ou & Penman	1989, JAE	Compustat	Annual
82	secured	Secured debt	Valta	2016, JFQA	Compustat	Annual
83	securedind	Secured debt indicator	Valta	2016, JFQA	Compustat	Annual
84	sgr	Sales growth	Lakonishok, Shleifer & Vishny	1994, JF	Compustat	Annual
85	sin	Sin stocks	Hong & Kacperczyk	2009, JFE	Compustat	Annual
86	sp	Sales to price	Barbee, Mukherji, & Raines	1996, FAJ	Compustat	Annual
87	std_dolvol	Volatility of liquidity (dollar trading volume)	Chordia, Subrahmanyam & Anshuman	2001, JFE	CRSP	Monthly
88	std_turn	Volatility of liquidity (share turnover)	Chordia, Subrahmanyam, &Anshuman	2001, JFE	CRSP	Monthly
89	stdacc	Accrual volatility	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Quarterly
90	stdcf	Cash flow volatility	Huang	2009, JEF	Compustat	Quarterly
91	tang	Debt capacity/firm tangibility	Almeida & Campello	2007, RFS	Compustat	Annual
92	tb	Tax income to book income	Lev & Nissim	2004, TAR	Compustat	Annual
93	turn	Share turnover	Datar, Naik & Radcliffe	1998, JFM	CRSP	Monthly
94	zerotrade	Zero trading days	Liu	2006, JFE	CRSP	Monthly

Table 1.15: Implied Sharpe Ratio Improvements

	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
Panel A: Common Factor Portfolios												
S&P 500	-	-	-	0.08	0.08	0.14	0.15	0.11	0.12	0.20	0.17	0.12
SMB	0.2	0.39	0.12	0.34	0.42	0.16	0.11	0.30	0.26	0.28	0.27	0.28
HML	0.12	0.09	0.20	0.09	0.15	0.17	0.04	0.20	0.21	0.18	0.20	0.20
RMW	-	0.15	0.06	-	-	-	-	0.9	0.06	0.11	0.07	0.07
CMA	0.10	-	0.00	-	0.14	-	-	0.20	0.18	0.12	0.20	0.15
UMD	-	-	-	0.12	-	0.27	-	-	-	0.06	0.08	0.10
Panel B: Sub-components of Factor Portfolios												
Big Value	0.05	0.00	-	0.03	0.07	0.14	0.12	0.09	0.10	0.15	0.13	0.11
Big Growth	-	-	-	0.08	0.06	0.14	0.13	0.11	0.12	0.16	0.13	0.12
Big Neutral	0.01	-	-	0.08	0.04	0.13	0.13	0.15	0.13	0.17	0.18	0.14
Small Value	0.02	0.15	0.10	0.06	0.08	0.11	0.05	0.14	0.13	0.14	0.13	0.12
Small Growth	-	0.03	-	-	-	0.14	0.21	0.01	0.09	0.10	0.08	0.10
Small Neutral	-	0.06	0.02	0.02	0.03	0.09	0.04	0.06	0.06	0.07	0.06	0.07
Big Conservative	-	-	-	0.09	0.04	0.10	0.05	0.11	0.10	0.14	0.12	0.10
Big Aggressive	-	-	-	0.04	0.09	0.20	0.23	0.16	0.18	0.21	0.18	0.17
Big Neutral	-	-	-	0.08	0.05	0.11	0.08	0.08	0.08	0.14	0.14	0.11
Small Conservative	-	0.12	0.08	0.00	0.04	0.10	0.06	0.09	0.09	0.10	0.09	0.09
Small Aggressive	-	0.09	0.00	-	0.03	0.16	0.22	0.06	0.11	0.12	0.10	0.12
Small Neutral	-	0.04	0.01	0.04	0.03	0.07	0.00	0.06	0.06	0.08	0.06	0.07
Big Robust	-	-	-	0.06	0.04	0.11	0.03	0.08	0.08	0.13	0.10	0.08
Big Weak	0.03	0.15	0.12	0.10	0.12	0.14	0.19	0.19	0.19	0.21	0.17	0.17
Big Neutral	-	-	-	0.06	0.02	0.14	0.12	0.11	0.13	0.15	0.15	0.13
Small Robust	-	0.04	-	0.00	-	0.07	0.02	0.02	0.05	0.06	0.05	0.05
Small Weak	0.04	0.17	0.11	-	0.08	0.17	0.22	0.13	0.15	0.16	0.15	0.15
Small Neutral	-	0.01	-	-	-	0.06	-	0.01	0.03	0.04	0.03	0.04
Big Up	-	-	-	0.06	0.11	0.11	0.08	0.07	0.07	0.10	0.10	0.09
Big Down	-	-	-	0.05	-	0.13	0.08	0.04	0.08	0.12	0.10	0.10
Big Medium	-	-	-	0.13	-	0.22	0.25	0.19	0.20	0.24	0.22	0.18
Small Up	-	0.08	0.06	-	0.03	0.07	0.00	0.01	0.01	0.02	0.02	0.03
Small Down	-	0.03	-	0.03	0.00	0.23	0.22	0.13	0.14	0.17	0.15	0.16
Small Medium	0.01	0.08	0.02	0.06	0.04	0.12	0.11	0.11	0.11	0.12	0.10	0.10

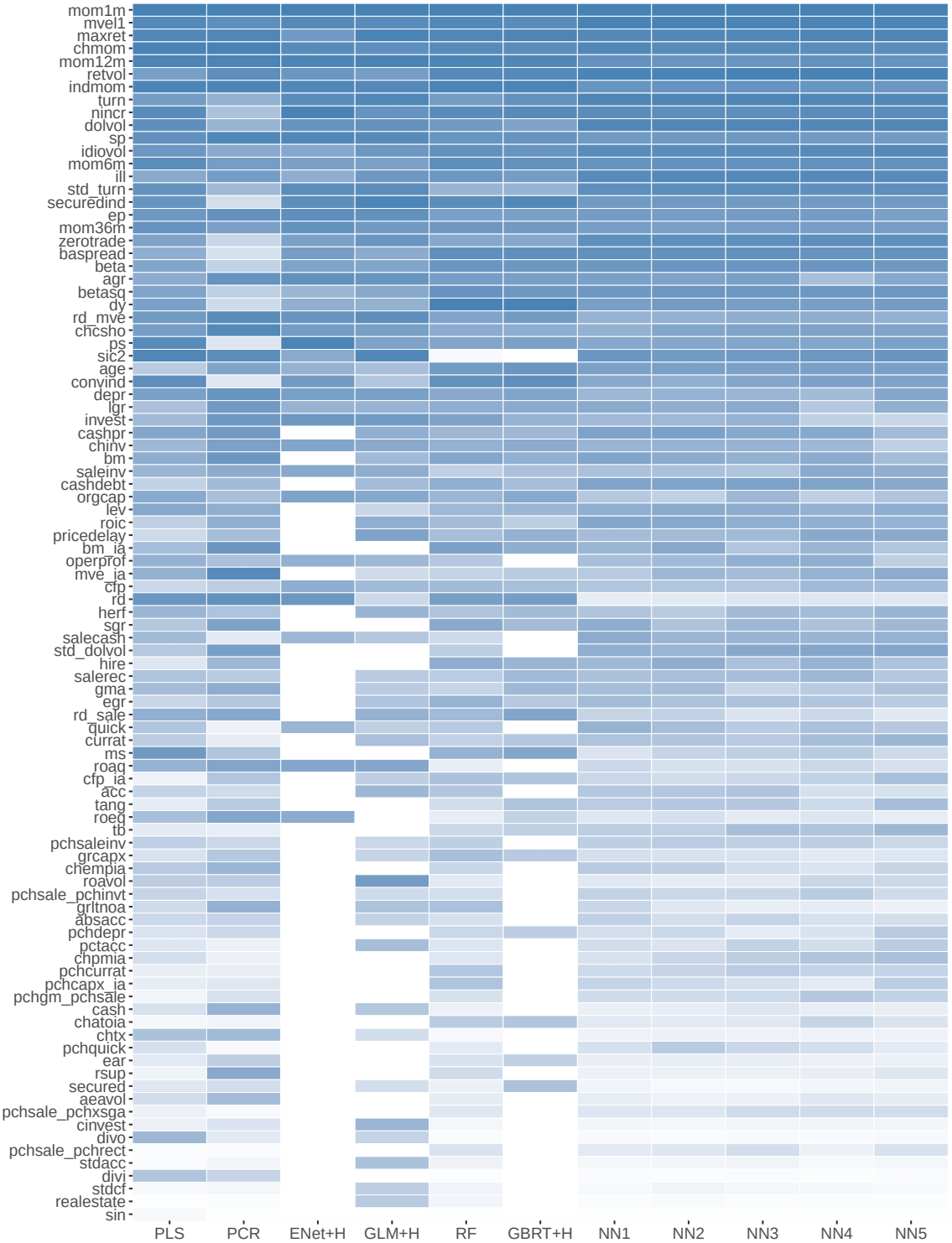
Note: Improvement in annualized Sharpe ratio ($SR^* - SR$) implied by the full sample Sharpe ratio of each portfolio together with machine learning predictive R^2_{oos} from Table 1.5. Cases with negative R^2_{oos} imply a Sharpe ratio deterioration and are omitted.

Figure 1.10: Characteristic Importance over Time by NN3



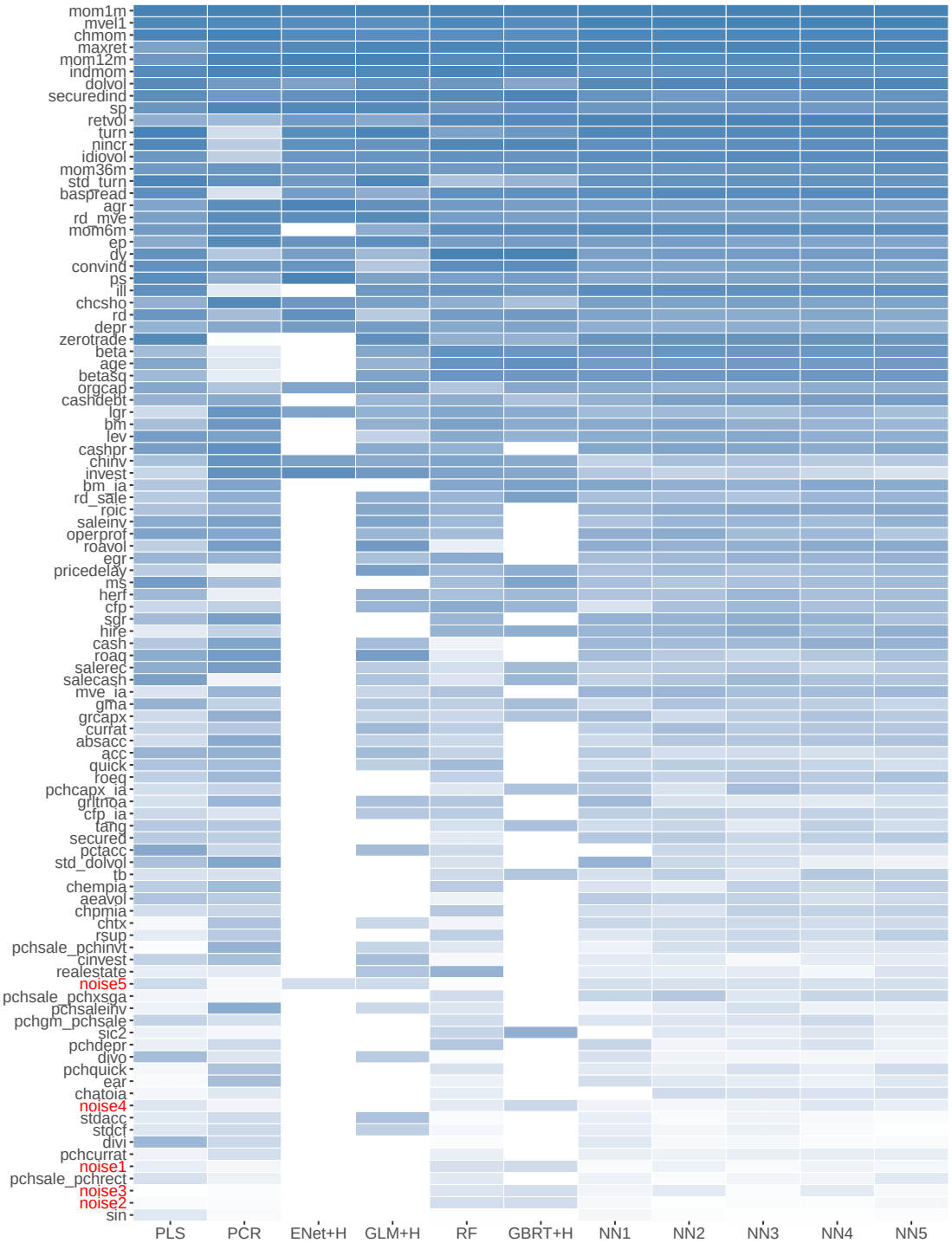
Note: This figure describes how NN3 ranks the 94 stock-level characteristics and the industry dummy (sic2) in terms of overall model contribution over 30 recursing training. Columns correspond to the year end of each of the 30 samples, and color gradients within each column indicate the most influential (dark blue) to least influential (white) variables. Characteristics are sorted in the same order of Figure 1.5.

Figure 1.11: Variable Importance Using SSD of Dimopoulos et al. [1995]



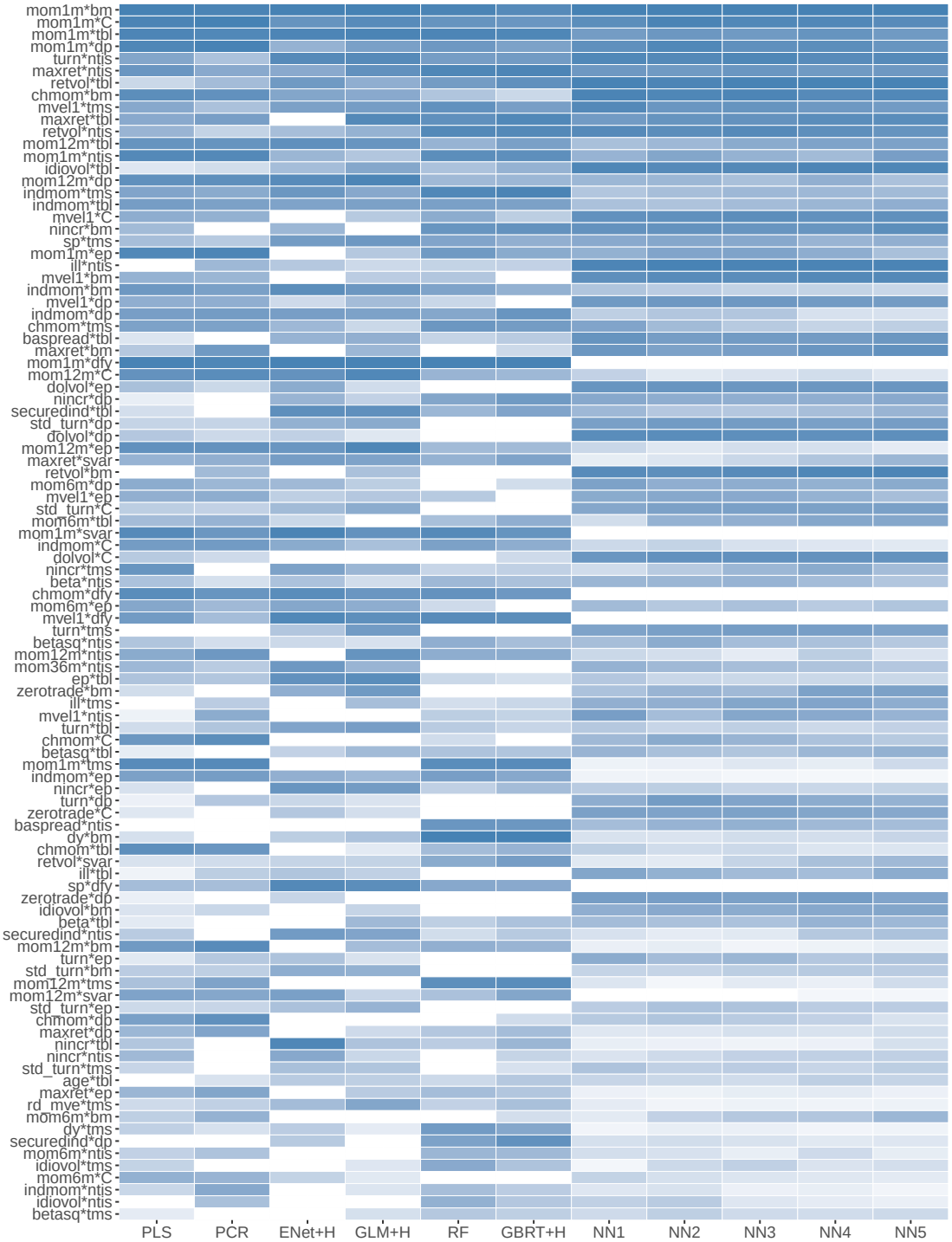
Note: Rankings of 94 stock-level characteristics and the industry dummy (sic2) in terms of SSD. Characteristics are ordered based on the sum of their ranks over all models, with the most influential characteristics on top and least influential on bottom. Columns correspond to individual models, and color gradients within each column indicate the most influential (dark blue) to least influential (white) variables.

Figure 1.12: Characteristic Importance with Placebo Variables



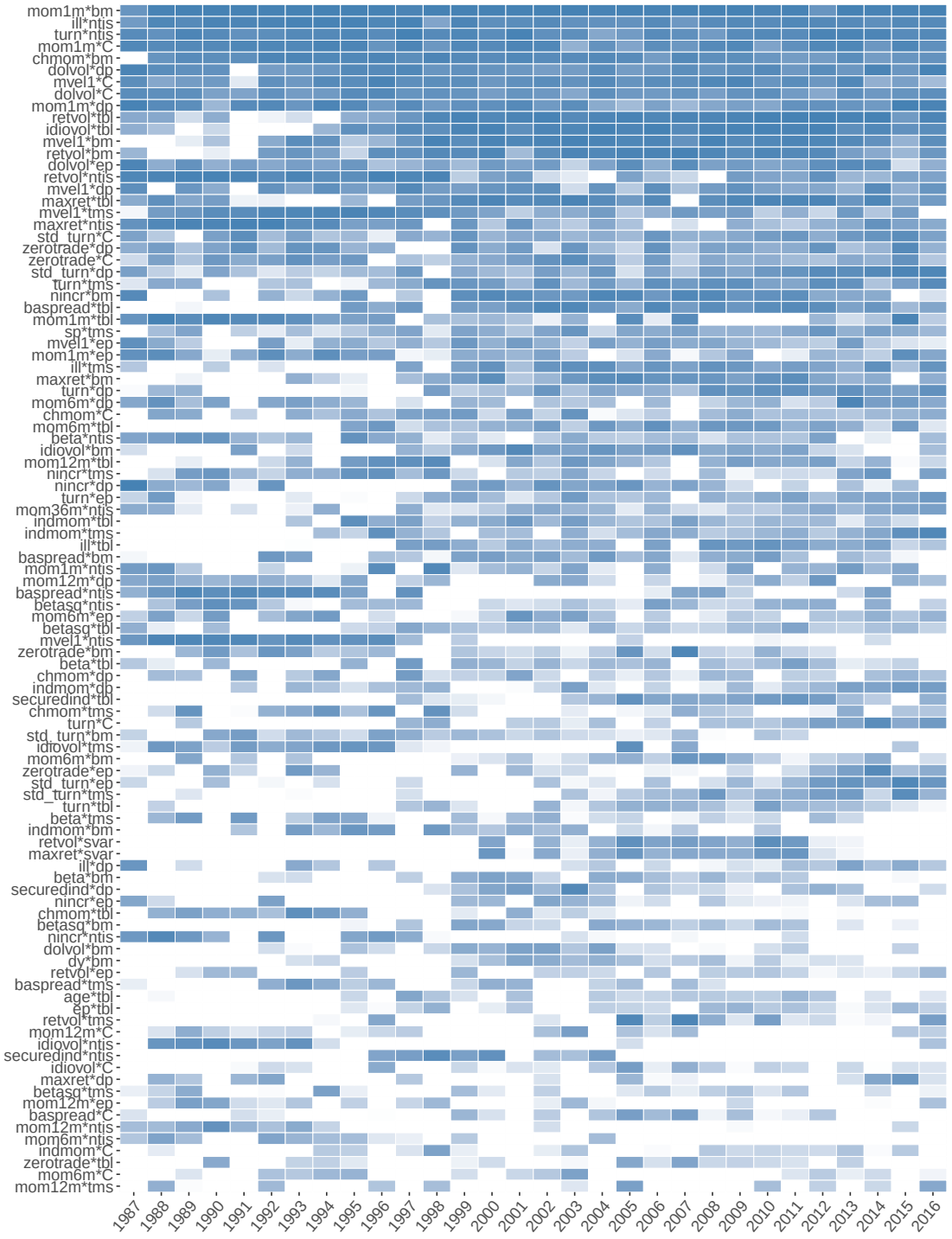
Note: This figure describes how each model ranks the 94 stock-level characteristics, the industry dummy (sic2), and five placebos in terms of overall model contribution. Columns correspond to individual models, and color gradients within each column indicate the most influential (dark blue) to least influential (white) variables. Characteristics are ordered based on the sum of their ranks over all models, with the most influential characteristics on top and least influential on bottom.

Figure 1.13: Stock/Macroeconomic Interactions



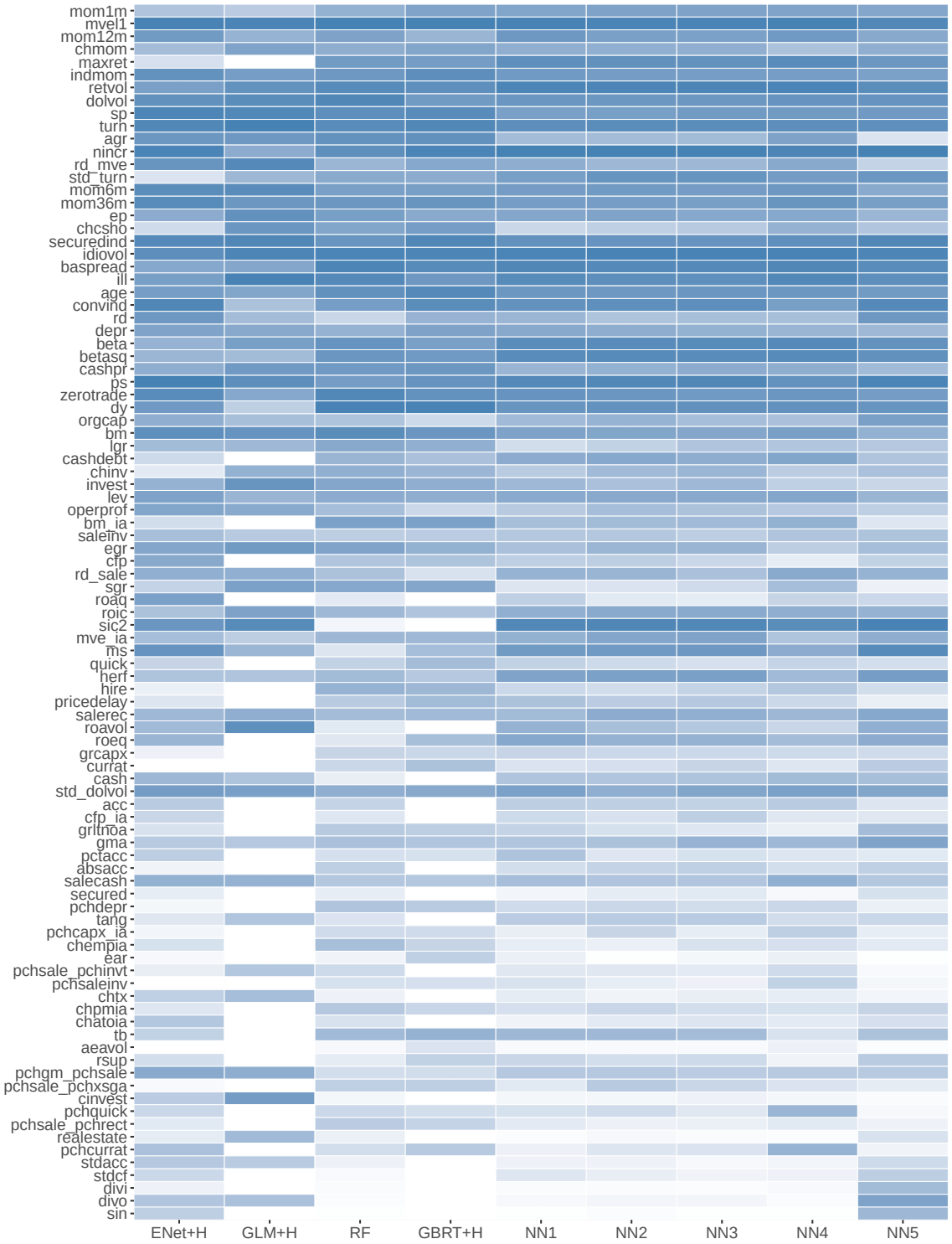
Note: Rankings of top 100 interactions between 94 stock-level stock characteristics and nine macro variables (including a constant, denoted C). Interactions are ordered based on the sum of their ranks over all models, with the most influential characteristics on top and least influential on bottom. Columns correspond to individual models, and color gradients within each column indicate the most influential (dark blue) to least influential (white) interactions.

Figure 1.14: Time Variation in Stock/Macroeconomic Interactions



Note: Rankings of top 100 interactions between 94 stock-level stock characteristics and nine macro variables (including a constant, denoted C). The list of top 100 interactions is based on the analysis in Figure 1.13. Color gradients indicate the most influential (dark blue) to least influential (white) interactions in the NN3 model in each training sample (the horizontal axis corresponds to the last year in each training sample).

Figure 1.15: Characteristic Importance at Annual Horizon



Note: This figure describes how each model ranks the 94 stock-level characteristics and the industry dummy (sic2) in terms of overall model contribution. Columns correspond to individual models, and color gradients within each column indicate the most influential (dark blue) to least influential (white) variables. Characteristics are sorted in the same order of Figure 1.5. The results are based on prediction at the annual horizon.

Table 1.16: Annual 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
Panel A: Common Factor Portfolios												
S&P 500	-4.90	0.43	-7.17	0.26	2.07	8.80	7.28	9.99	12.02	15.68	15.30	13.15
SMB	3.77	4.23	8.26	4.22	6.96	6.54	4.27	0.05	1.31	2.59	4.33	4.45
HML	3.01	-0.52	4.08	-0.15	6.33	7.02	2.17	9.14	8.09	7.86	3.97	3.63
RMW	4.66	17.11	6.67	1.19	3.45	3.51	5.31	7.03	5.03	3.58	0.61	0.90
CMA	4.50	-1.52	7.94	-9.01	7.69	1.73	-8.36	5.89	7.27	0.93	-7.18	-8.11
UMD	-27.52	-12.44	-5.62	-16.27	-8.06	-7.57	-8.29	-12.78	-8.71	-7.35	-6.45	-6.74
Panel B: Sub-components of Factor Portfolios												
Big Value	1.83	4.88	-4.04	3.62	0.49	9.50	5.86	8.76	8.54	12.42	9.95	7.56
Big Growth	-12.06	-6.92	-10.22	-2.13	2.44	7.14	6.93	7.47	11.06	11.67	13.37	10.03
Big Neutral	-3.83	3.09	-6.58	1.19	1.24	8.52	6.91	8.31	11.51	14.60	12.92	9.95
Small Value	4.31	10.81	8.94	8.41	4.31	8.05	3.75	7.24	6.37	7.48	6.60	4.81
Small Growth	2.49	2.87	3.19	0.21	0.03	6.20	2.13	3.96	5.52	6.84	2.60	7.23
Small Neutral	-1.52	5.21	2.10	2.29	2.29	4.18	1.78	6.46	5.55	6.68	3.69	6.14
Big Conservative	-10.42	-2.42	-9.77	-3.77	5.17	8.44	5.26	-1.31	8.64	9.65	12.47	6.09
Big Aggressive	-1.65	1.89	-4.72	1.36	2.00	7.42	6.67	11.00	11.74	13.08	11.27	10.67
Big Neutral	-9.18	-1.62	-9.42	2.03	2.43	9.62	8.39	10.88	13.03	15.61	15.75	13.56
Small Conservative	-0.38	6.36	5.01	3.19	2.35	4.60	0.62	5.31	5.39	5.97	4.22	4.71
Small Aggressive	3.33	5.12	2.88	1.04	0.37	6.43	3.23	2.50	4.50	5.50	1.47	6.56
Small Neutral	-0.53	5.84	3.52	4.46	3.59	7.08	2.96	8.41	7.13	8.68	5.77	8.47
Big Robust	-7.53	-2.55	-9.18	1.33	5.42	7.61	6.60	12.55	12.04	13.92	15.29	13.35
Big Weak	-3.40	3.09	-7.15	-1.02	-1.12	9.62	7.62	4.41	9.95	11.39	11.73	8.40
Big Neutral	-4.17	5.46	-4.57	-3.18	-2.12	6.24	4.47	4.18	6.23	9.47	3.70	2.95
Small Robust	-2.37	0.93	-0.20	0.76	3.72	0.41	-0.87	2.92	3.67	4.47	0.86	4.19
Small Weak	3.88	9.89	5.68	2.15	-1.11	7.53	3.10	-0.48	1.53	2.96	1.61	1.08
Small Neutral	3.00	7.99	4.40	4.60	3.58	9.21	5.75	10.03	7.39	9.82	7.06	9.09
Big Up	-23.55	-11.77	-19.16	-5.11	0.52	6.15	6.21	4.26	11.44	11.11	14.48	10.62
Big Down	-4.66	0.39	-2.79	-0.15	0.71	7.64	5.53	3.58	8.78	9.54	10.32	6.79
Big Medium	6.26	10.24	7.36	6.25	3.83	7.73	5.38	8.74	9.61	11.36	9.96	6.22
Small Up	-6.68	3.82	0.71	-2.83	1.57	1.84	-0.19	-4.22	0.70	1.12	-1.42	2.83
Small Down	2.80	5.59	4.84	2.87	0.50	7.23	3.49	3.24	4.63	5.90	3.28	5.22
Small Medium	-2.92	-0.49	-1.70	-1.80	0.81	2.00	-0.40	-1.64	1.96	1.79	0.51	3.49

Note: In this table, we report the out-of-sample predictive R^2s for 30 portfolios using OLS with size, book-to-market, and 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), respectively. "+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 24 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 The results are based on prediction at the annual horizon.

Table 1.17: Performance of Machine Learning Portfolios (Equally Weighted)

	OLS-3+H				PLS				PCR			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.14	0.11	7.99	0.05	-0.83	-0.26	6.41	-0.14	-0.71	-0.65	7.04	-0.32
2	0.17	0.35	6.81	0.18	-0.20	0.19	5.92	0.11	-0.11	0.16	6.23	0.09
3	0.35	0.44	6.09	0.25	0.12	0.40	5.49	0.25	0.19	0.40	5.67	0.25
4	0.49	0.63	5.61	0.39	0.39	0.67	5.06	0.46	0.42	0.58	5.45	0.37
5	0.63	0.73	5.24	0.49	0.62	0.69	5.14	0.47	0.63	0.72	5.11	0.49
6	0.75	0.83	4.88	0.59	0.84	0.77	5.14	0.52	0.81	0.80	4.98	0.55
7	0.88	0.75	4.73	0.55	1.06	0.88	5.12	0.60	1.01	0.98	5.02	0.68
8	1.03	0.80	4.72	0.59	1.32	1.01	5.29	0.66	1.23	1.08	5.02	0.75
9	1.22	1.14	4.73	0.83	1.67	1.28	5.60	0.79	1.52	1.33	5.28	0.88
High(H)	1.60	1.45	5.21	0.96	2.38	1.82	6.16	1.02	2.12	1.81	5.93	1.06
H-L	1.73	1.34	5.59	0.83	3.21	2.08	4.89	1.47	2.83	2.45	4.51	1.89
	ENet+H				GLM+H				RF			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.04	-0.24	6.43	-0.13	-0.49	-0.50	6.81	-0.25	0.26	-0.48	7.16	-0.23
2	0.27	0.44	5.90	0.26	0.01	0.32	5.80	0.19	0.44	0.24	5.67	0.15
3	0.44	0.52	5.27	0.34	0.29	0.56	5.46	0.36	0.53	0.55	5.36	0.36
4	0.59	0.70	4.73	0.51	0.50	0.61	5.22	0.41	0.60	0.62	5.15	0.42
5	0.73	0.71	4.94	0.49	0.68	0.72	5.11	0.49	0.67	0.66	5.11	0.44
6	0.87	0.79	5.00	0.55	0.84	0.78	5.12	0.53	0.73	0.77	5.13	0.52
7	1.01	0.85	5.21	0.56	1.00	0.78	5.06	0.54	0.80	0.74	5.10	0.50
8	1.17	0.88	5.47	0.56	1.18	0.89	5.14	0.60	0.87	0.99	5.29	0.65
9	1.36	0.85	5.90	0.50	1.41	1.25	5.80	0.75	0.97	1.22	5.67	0.74
High(H)	1.72	1.86	7.27	0.89	1.89	1.81	6.57	0.96	1.20	1.90	7.03	0.94
H-L	1.76	2.11	5.50	1.33	2.38	2.31	4.41	1.82	0.94	2.38	5.57	1.48
	GBRT+H				NN1				NN2			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.49	-0.37	6.46	-0.20	-0.45	-0.78	7.43	-0.36	-0.32	-1.01	7.79	-0.45
2	-0.16	0.42	5.80	0.25	0.15	0.22	6.24	0.12	0.20	0.17	6.34	0.09
3	0.02	0.56	5.31	0.36	0.43	0.47	5.55	0.29	0.43	0.52	5.49	0.33
4	0.17	0.74	5.43	0.47	0.64	0.64	5.00	0.45	0.59	0.71	5.02	0.49
5	0.33	0.63	5.31	0.41	0.80	0.80	4.76	0.58	0.72	0.76	4.60	0.57
6	0.46	0.83	5.23	0.55	0.95	0.85	4.63	0.63	0.84	0.81	4.52	0.62
7	0.59	0.67	5.13	0.45	1.12	0.84	4.66	0.62	0.97	0.94	4.61	0.70
8	0.72	0.82	5.08	0.56	1.32	0.88	4.95	0.62	1.14	0.92	4.86	0.66
9	0.88	1.12	5.41	0.72	1.63	1.17	5.62	0.72	1.41	1.10	5.55	0.69
High(H)	1.19	1.77	6.69	0.92	2.43	2.13	7.34	1.00	2.25	2.30	7.81	1.02
H-L	1.68	2.14	4.28	1.73	2.89	2.91	4.72	2.13	2.57	3.31	4.92	2.33
	NN3				NN4				NN5			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.31	-0.92	7.94	-0.40	-0.19	-0.95	7.83	-0.42	-0.08	-0.83	7.92	-0.36
2	0.22	0.16	6.46	0.09	0.29	0.17	6.50	0.09	0.33	0.24	6.64	0.12
3	0.45	0.44	5.40	0.28	0.49	0.45	5.58	0.28	0.51	0.53	5.65	0.32
4	0.60	0.66	4.83	0.48	0.62	0.57	4.94	0.40	0.62	0.59	4.91	0.41
5	0.73	0.77	4.58	0.58	0.72	0.70	4.57	0.53	0.71	0.68	4.56	0.51
6	0.85	0.81	4.47	0.63	0.81	0.75	4.42	0.59	0.80	0.76	4.43	0.60
7	0.97	0.86	4.62	0.64	0.91	0.86	4.47	0.67	0.88	0.88	4.60	0.66
8	1.12	0.93	4.82	0.67	1.04	1.06	4.82	0.76	1.01	0.95	4.90	0.67
9	1.38	1.18	5.51	0.74	1.28	1.24	5.57	0.77	1.25	1.17	5.60	0.73
High(H)	2.28	2.35	8.11	1.00	2.16	2.37	8.03	1.02	2.08	2.27	7.95	0.99
H-L	2.58	3.27	4.80	2.36	2.35	3.33	4.71	2.45	2.16	3.09	4.98	2.15

Note: Performance of equal-weight decile portfolios sorted on out-of-sample machine learning return forecasts. "Pred", "Avg", "Std", and "SR" report the predicted monthly returns for each decile, the average realized monthly returns, their realized standard deviations, and annualized Sharpe ratios, respectively.

Table 1.18: Performance of Machine Learning Portfolios (Equally Weighted, Excluding Microcaps)

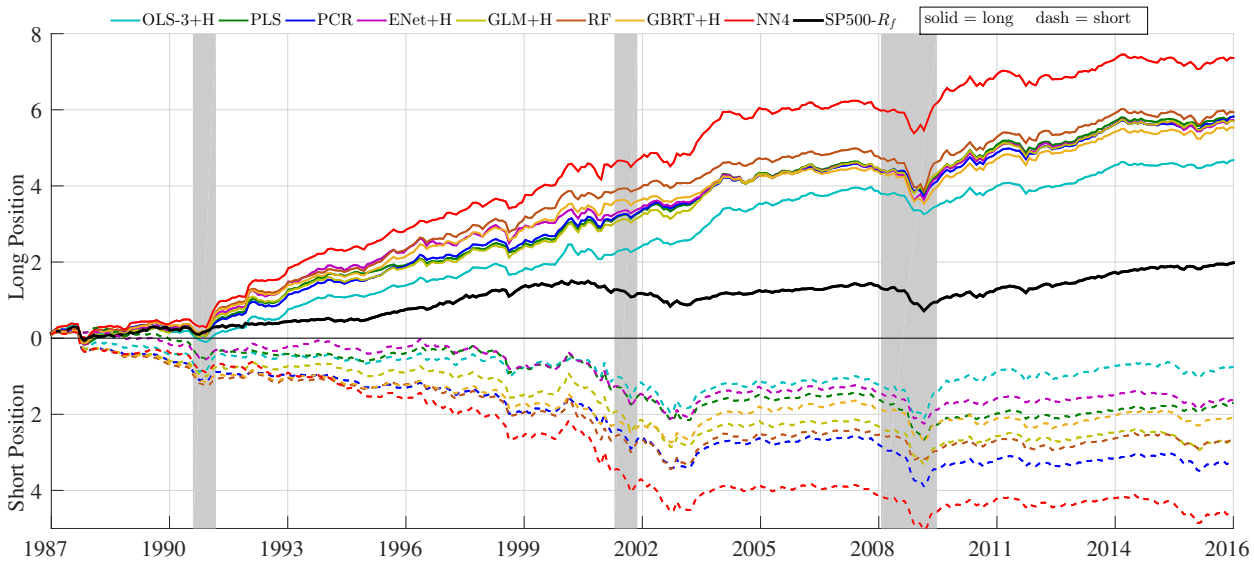
	OLS-3+H				PLS				PCR			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.17	0.00	7.97	0.00	-0.88	-0.33	6.59	-0.17	-0.72	-0.50	7.04	-0.25
2	0.12	0.19	6.53	0.10	-0.26	0.27	5.83	0.16	-0.13	0.16	6.14	0.09
3	0.31	0.40	5.72	0.24	0.06	0.35	5.41	0.22	0.16	0.36	5.52	0.22
4	0.45	0.52	5.32	0.34	0.31	0.54	5.16	0.36	0.39	0.52	5.21	0.35
5	0.58	0.63	4.96	0.44	0.54	0.66	5.01	0.46	0.59	0.63	4.94	0.44
6	0.70	0.63	4.71	0.46	0.75	0.70	4.97	0.49	0.77	0.71	4.83	0.51
7	0.82	0.66	4.64	0.49	0.96	0.82	4.71	0.60	0.96	0.76	4.80	0.55
8	0.96	0.75	4.70	0.56	1.21	0.85	5.12	0.57	1.17	0.95	4.84	0.68
9	1.15	1.04	4.95	0.73	1.53	1.02	5.32	0.66	1.46	1.09	5.14	0.74
High(H)	1.47	1.33	5.35	0.86	2.21	1.33	5.87	0.78	2.03	1.47	5.83	0.87
H-L	1.64	1.32	5.66	0.81	3.09	1.66	4.69	1.22	2.75	1.97	4.61	1.48
	ENet+H				GLM+H				RF			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.05	-0.23	6.51	-0.12	-0.51	-0.35	6.81	-0.18	0.27	-0.43	7.03	-0.21
2	0.25	0.42	5.72	0.26	-0.03	0.32	5.71	0.20	0.44	0.23	5.58	0.15
3	0.42	0.53	5.14	0.36	0.25	0.54	5.34	0.35	0.52	0.50	5.19	0.33
4	0.56	0.60	4.82	0.43	0.45	0.59	5.12	0.40	0.59	0.58	5.04	0.40
5	0.69	0.69	4.80	0.50	0.63	0.65	4.98	0.45	0.66	0.58	4.97	0.41
6	0.82	0.73	4.89	0.52	0.79	0.68	4.96	0.48	0.72	0.65	5.04	0.45
7	0.96	0.83	4.74	0.61	0.95	0.70	4.91	0.49	0.78	0.65	4.99	0.45
8	1.11	0.77	5.31	0.50	1.12	0.75	4.95	0.53	0.85	0.85	5.02	0.58
9	1.30	0.78	5.74	0.47	1.34	0.95	5.30	0.62	0.92	1.08	5.34	0.70
High(H)	1.65	1.04	6.78	0.53	1.79	1.31	6.33	0.72	1.09	1.43	6.65	0.74
H-L	1.70	1.27	4.90	0.90	2.30	1.65	4.44	1.29	0.81	1.86	5.25	1.22
	GBRT+H				NN1				NN2			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.47	-0.28	6.25	-0.15	-0.47	-0.76	7.48	-0.35	-0.33	-0.92	8.00	-0.40
2	-0.15	0.38	5.55	0.24	0.12	0.20	6.36	0.11	0.19	0.20	6.51	0.10
3	0.02	0.52	5.22	0.34	0.40	0.48	5.54	0.30	0.41	0.55	5.63	0.34
4	0.17	0.67	5.31	0.44	0.59	0.63	5.01	0.43	0.56	0.70	5.03	0.48
5	0.32	0.55	5.24	0.36	0.74	0.72	4.76	0.53	0.68	0.74	4.59	0.56
6	0.45	0.76	4.95	0.54	0.87	0.85	4.61	0.64	0.79	0.84	4.49	0.65
7	0.57	0.52	5.10	0.35	1.01	0.87	4.60	0.65	0.89	0.90	4.51	0.69
8	0.69	0.70	4.90	0.50	1.16	0.85	4.68	0.63	1.02	0.93	4.69	0.68
9	0.84	1.02	5.26	0.67	1.38	1.00	5.13	0.68	1.19	0.96	4.99	0.67
High(H)	1.10	1.30	6.25	0.72	1.91	1.29	6.25	0.72	1.68	1.26	6.22	0.70
H-L	1.57	1.58	3.86	1.42	2.38	2.05	4.50	1.58	2.01	2.18	4.74	1.60
	NN3				NN4				NN5			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.31	-0.82	8.18	-0.35	-0.19	-0.87	8.05	-0.38	-0.08	-0.75	8.11	-0.32
2	0.20	0.16	6.55	0.08	0.28	0.23	6.68	0.12	0.32	0.22	6.75	0.12
3	0.43	0.46	5.51	0.29	0.47	0.45	5.61	0.28	0.49	0.51	5.70	0.31
4	0.57	0.66	4.86	0.47	0.59	0.65	4.93	0.45	0.61	0.58	4.98	0.40
5	0.69	0.76	4.63	0.57	0.68	0.65	4.60	0.49	0.69	0.69	4.55	0.52
6	0.79	0.79	4.44	0.61	0.76	0.71	4.48	0.55	0.76	0.76	4.43	0.60
7	0.89	0.87	4.48	0.67	0.84	0.90	4.45	0.70	0.83	0.84	4.45	0.65
8	1.01	0.91	4.71	0.67	0.94	0.92	4.59	0.70	0.91	0.92	4.70	0.68
9	1.17	1.00	5.02	0.69	1.07	1.13	5.00	0.78	1.04	1.02	5.10	0.69
High(H)	1.64	1.37	6.34	0.75	1.52	1.39	6.37	0.75	1.48	1.36	6.34	0.74
H-L	1.95	2.19	4.84	1.57	1.70	2.26	4.63	1.69	1.56	2.11	4.95	1.48

Table 1.19: OLS Benchmark Models

	R^2		Sharpe Ratio		
Model	Stock	S&P 500	Equal-weight	Value-weight	Description
OLS-3	0.16	-0.22	0.83	0.61	mom12m, size, bm
OLS-7	0.18	0.24	1.12	0.74	OLS-3 plus acc, roaq, agr, egr
OLS-15	0.19	0.68	1.15	0.86	OLS-7 plus dy, mom36m, beta, retvol, turn, lev, sp
RF	0.33	1.37	1.48	0.98	
NN3	0.40	1.80	2.36	1.20	

Note: In this table, we report the out-of-sample performance of three different OLS benchmark models recommended by Lewellen [2015] with either three, seven, or 15 predictors. We report predictive R^2 for the stock-level panel and the S&P 500 index. We report long-short decile spread Sharpe ratios with equal-weight and value-weight formation. For comparison, we also report the performance the NN3 and random forest models.

Figure 1.16: Cumulative Return of Machine Learning Portfolios (Equally Weighted)



Note: Cumulative log returns of portfolios sorted on out-of-sample machine learning return forecasts. The solid and dash lines represent long (top decile) and short (bottom decile) positions, respectively. The shaded periods show NBER recession dates. All portfolios are equally weighted.

1.5.7 Updated Firm Characteristics Data Comparison

The updated firm characteristics dataset contains samples from 195703 to 202012 (the old version data is from 195703 to 201612). We select the overlapping period (195703-201612) to do the comparison.

- Sample size:
  NEW =3760033, OLD=3760208
  In NEW but not in OLD=3256
  In OLD but not in NEW=3431
  The difference of sample size results from some minor permnos because of the WRDS database change. (e.g. [15524, 15631 15656, 15913, 16306 81021, 81628, 85226, 87032, 88467, 91953, 93263], and most are from the financial sector)

- *bm* and *bm_ia*:

  We updated the calculation formulas of *bm* and *bm_ia*.

$$corr(\text{old } bm, \text{new } bm) = 0.9096$$

$$corr(crs\_rank(\text{old } bm), crs\_rank(\text{new } bm)) = 0.9752.$$

  Most of samples have the exactly same values of *bm*. For the characteristic *bm_ia*,

$$corr(crs\_rank(\text{old } bm\_ia), crs\_rank(\text{new } bm\_ia)) = 0.9267.$$

  Since we do cross-sectional ranking for all firm characteristics before training, we only care about the ranking change of new bm and old bm. We can see the correlations of *bm* and *bm_ia* are very high in the old and new version of our dataset.

- Other characteristics:

  We check the correlations of other firm characteristics between new version and old version dataset. The average corr of 92 firm characteristics (exclude *bm* and *bm_ia*) = 0.998. So that other firm characteristics are very close to the old version.

- Training with the updated data (the same OOS period)

  We just reported the main performance of the Machine learning methods. We use the same parameters and the same random seeds for training, and check the performance in the same 30-years OOS perios (198701-201612). We can see all the methods get very close R^2s and sharp ratios (the simple OLS has the largest gap). If we check the

Table 1.20: 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
Old Version	-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
New Version	-3.38	0.16	0.26	0.26	0.12	0.20	0.34	0.34	0.32	0.39	0.40	0.39	0.35

Table 1.21: Monthly out-of-sample stock-level prediction performance (value-weighted SR)

	OLS +H	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
Old Version	-	0.61	0.72	0.88	0.39	0.76	0.98	0.81	1.17	1.16	1.20	1.35	1.15
New Version	-	0.61	0.70	0.88	0.40	0.79	0.96	0.83	1.15	1.16	1.20	1.32	1.13

correlations of predicted returns between old and new version of data, the average corr value is 0.985.

- Other findings

  If we only care about the H-L portfolios, the cosine-similarity is a better loss function than MSE. Because it is not influenced by outliers with huge returns, cosine-similarity (or named as IC) is used in all industry trading firms. Also LSTM and Transformers can get much higher SR/returns as well. LSTM can learn the historical information of the firm characteristics.

1.5.8 201701-202012 Out-of-Sample Performance of All Models

We do the exactly same empirical analysis using the updated data (including firm characteristic data and macroeconomic data). For the new 48 months (201701-202012) in OOS period, we still implement recursive strategy when training/validation/testing. The prediction performance (percentage R^2) and Sharp ratios of long-short portfolios are reported in the following tables. The cumulative returns plots are also attached.

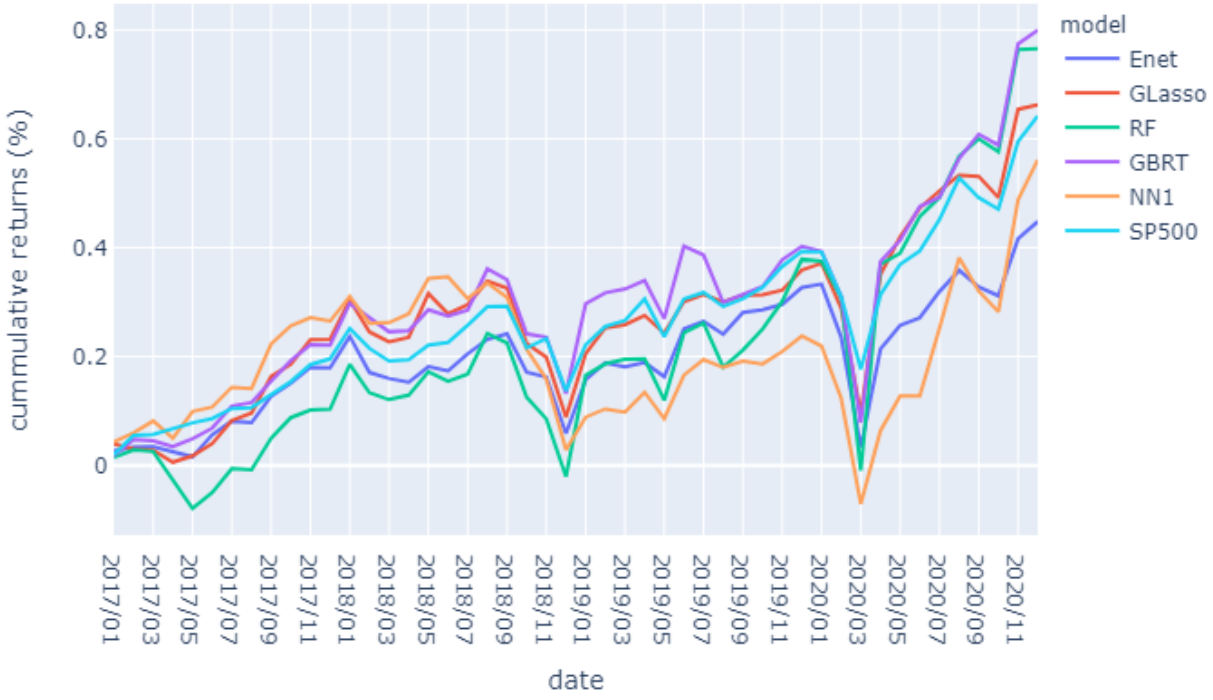
We can see that the margin of machine learning techniques becomes weak in the recent 4 years. There is even no benefit if the aggressive investors trade based on the return prediction of ML models, compared with the passive investors who buy and hold the market.

Table 1.22: 201701-202012 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	-2.55	0.20	0.26	0.10	0.33	0.31	0.25	0.21	0.21	0.23	0.24	0.25	0.26
Top 1,000	-7.42	0.79	1.11	0.52	1.27	0.79	0.97	0.90	1.05	1.07	1.09	1.12	1.11
Bottom 1,000	-1.00	0.11	0.12	0.03	0.17	0.19	0.13	0.10	0.10	0.10	0.10	0.10	0.12

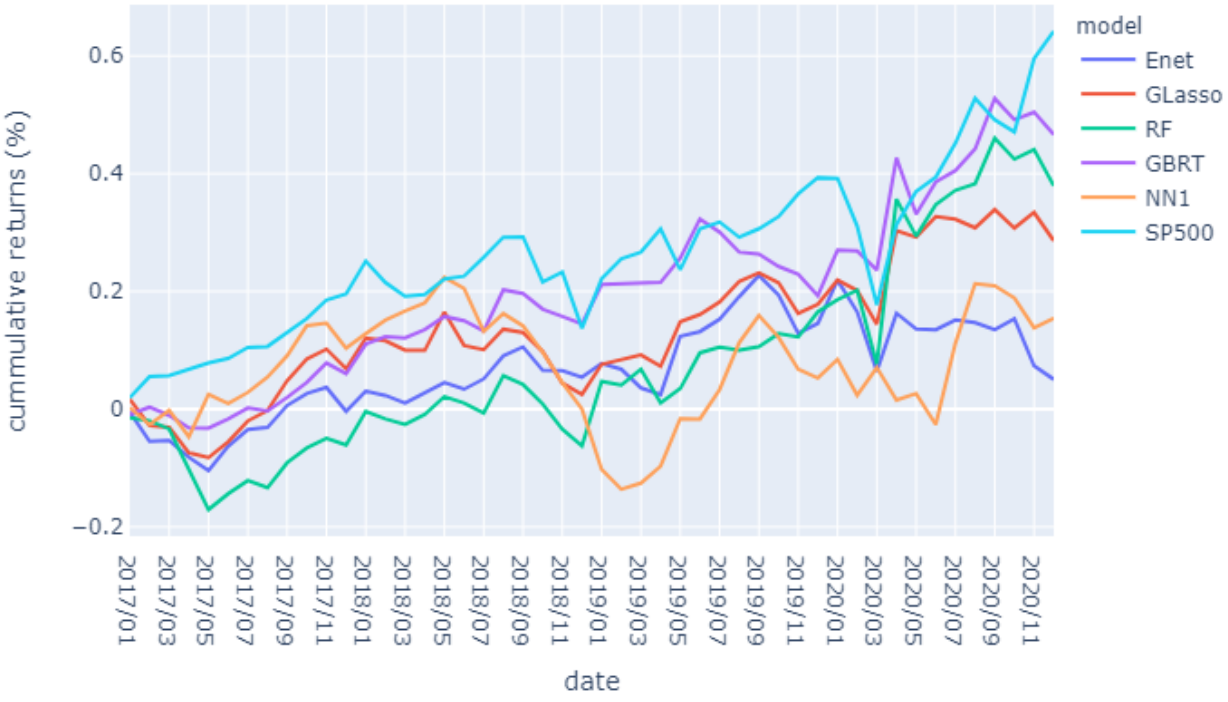
In this table, we report monthly R^2_{oos} for the entire panel of stocks using OLS with all variables (OLS), OLS using only size, book-to-market, and momentum (OLS-3), PLS, PCR, elastic net (ENet), generalize linear model (GLM), random forest (RF), gradient boosted regression trees (GBRT), and neural 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.

Figure 1.17: Cumulative Return of Machine Learning Portfolios (Value Weighted, Long-side, 201701-202012)



Note: Cumulative log returns of portfolios (long-side) sorted on out-of-sample machine learning return forecasts. All portfolios are value weighted.

Figure 1.18: Cumulative Return of Machine Learning Portfolios (Value Weighted, Long-Short, 201701-202012)



Note: Cumulative log returns of portfolios (long-side minus short side) sorted on out-of-sample machine learning return forecasts. All portfolios are value weighted.

Table 1.23: 201701-202012 Performance of Machine Learning Portfolios (Equally Weighted)

	OLS-3+H				PLS				PCR			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.02	1.11	9.52	0.40	0.63	1.10	6.30	0.61	-0.02	0.66	7.00	0.33
2	0.26	1.13	8.35	0.47	0.54	0.56	6.11	0.32	0.31	1.06	6.52	0.56
3	0.41	1.07	7.37	0.50	0.45	0.05	3.39	0.05	0.45	1.10	6.06	0.63
4	0.54	0.98	6.35	0.53	0.48	0.82	4.45	0.64	0.55	0.94	5.76	0.56
5	0.66	0.86	6.00	0.50	0.60	0.85	6.42	0.46	0.63	0.86	5.59	0.54
6	0.77	0.93	5.40	0.59	0.57	1.03	4.55	0.78	0.73	0.88	6.21	0.49
7	0.88	0.90	5.08	0.61	0.54	0.52	6.33	0.29	0.85	0.98	6.48	0.53
8	1.00	0.82	5.31	0.53	0.65	1.30	5.48	0.83	0.99	1.06	6.71	0.55
9	1.15	1.08	5.72	0.66	0.63	0.75	6.42	0.41	1.17	1.21	7.56	0.56
High(H)	1.45	1.17	6.41	0.63	0.62	0.43	5.78	0.26	1.51	1.27	7.82	0.56
H-L	1.43	0.06	4.26	0.05	-0.01	-0.67	3.16	-0.73	1.53	0.61	4.15	0.51
	ENet+H				GLM+H				RF			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.37	0.60	6.02	0.35	0.04	0.34	6.21	0.19	0.53	0.62	5.99	0.36
2	0.51	0.77	5.30	0.50	0.32	0.71	5.87	0.42	0.55	0.32	5.44	0.21
3	0.59	0.68	5.56	0.42	0.47	0.71	5.97	0.41	0.65	0.70	5.90	0.41
4	0.66	0.96	6.15	0.54	0.57	0.67	6.04	0.38	0.69	0.93	5.62	0.57
5	0.73	0.97	6.24	0.54	0.67	1.06	6.04	0.61	0.71	1.29	6.02	0.74
6	0.79	1.09	6.92	0.55	0.76	1.10	6.19	0.62	0.73	1.36	6.11	0.77
7	0.86	1.18	6.92	0.59	0.87	1.02	6.82	0.52	0.75	0.77	6.65	0.40
8	0.93	1.11	7.01	0.55	1.01	1.29	6.96	0.64	0.78	1.21	6.55	0.64
9	1.02	1.33	7.35	0.63	1.21	1.59	7.55	0.73	0.81	1.14	7.42	0.53
High(H)	1.18	1.34	8.11	0.57	1.69	1.54	8.38	0.64	0.92	1.81	10.59	0.59
H-L	0.81	0.74	4.24	0.60	1.65	1.20	4.43	0.94	0.39	1.19	7.15	0.58
	GBRT+H				NN1				NN2			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.30	0.58	6.21	0.32	-0.02	0.72	7.53	0.33	0.33	0.71	7.68	0.32
2	0.36	0.63	5.78	0.38	0.32	1.02	6.79	0.52	0.52	1.27	7.24	0.61
3	0.49	0.62	6.04	0.36	0.47	0.82	6.39	0.44	0.60	0.94	6.53	0.50
4	0.55	0.99	6.57	0.52	0.59	0.90	5.99	0.52	0.66	0.80	5.85	0.47
5	0.60	0.96	5.92	0.56	0.68	0.91	5.87	0.54	0.70	0.95	5.70	0.58
6	0.63	1.12	5.99	0.65	0.77	0.95	5.89	0.56	0.75	0.88	5.69	0.53
7	0.67	1.14	5.80	0.68	0.86	0.99	5.99	0.57	0.79	1.07	5.92	0.63
8	0.70	1.17	6.42	0.63	0.95	1.03	6.16	0.58	0.83	1.09	6.18	0.61
9	0.75	1.37	7.25	0.66	1.08	1.06	6.55	0.56	0.89	0.94	6.50	0.50
High(H)	0.86	1.62	9.61	0.58	1.43	1.63	8.51	0.67	1.08	1.37	8.39	0.57
H-L	0.55	1.05	5.49	0.66	1.45	0.91	3.83	0.82	0.75	0.66	3.56	0.64
	NN3				NN4				NN5			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.55	0.81	7.66	0.37	0.62	0.78	7.99	0.34	0.70	0.79	7.65	0.36
2	0.64	1.07	7.35	0.50	0.69	1.17	7.47	0.54	0.72	0.88	6.90	0.44
3	0.68	0.86	6.59	0.45	0.71	0.84	6.76	0.43	0.73	0.86	6.29	0.47
4	0.70	0.75	6.15	0.42	0.73	0.71	6.17	0.40	0.74	1.00	6.03	0.57
5	0.72	0.86	5.57	0.53	0.75	0.94	5.71	0.57	0.75	0.80	5.53	0.50
6	0.74	1.08	5.60	0.67	0.76	0.91	5.66	0.56	0.75	0.97	5.81	0.58
7	0.77	1.06	5.95	0.62	0.77	1.11	5.78	0.67	0.75	1.07	5.69	0.65
8	0.79	1.15	6.12	0.65	0.78	1.03	5.80	0.62	0.76	1.07	5.98	0.62
9	0.82	0.97	6.41	0.53	0.80	1.03	6.25	0.57	0.77	1.01	6.62	0.53
High(H)	0.93	1.43	8.44	0.59	0.89	1.52	8.34	0.63	0.81	1.61	9.40	0.59
H-L	0.38	0.61	3.83	0.55	0.27	0.75	3.82	0.68	0.11	0.83	4.17	0.69

Note: In this table, we report the performance of prediction-sorted portfolios over the 4-year out-of-sample testing period (201701-202012). All portfolios are equally weighted.

Table 1.24: 201701-202012 Performance of Machine Learning Portfolios (Value Weighted)

	OLS-3+H				PLS				PCR			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	-0.01	1.15	7.03	0.57	0.63	1.07	4.84	0.77	0.01	1.26	5.13	0.85
2	0.26	0.85	5.49	0.54	0.54	0.54	4.73	0.40	0.31	1.27	5.16	0.86
3	0.41	1.06	5.33	0.69	0.45	0.25	3.10	0.28	0.45	1.21	5.29	0.79
4	0.54	1.18	4.59	0.89	0.48	0.82	3.69	0.77	0.55	0.94	5.10	0.64
5	0.66	1.14	4.72	0.84	0.60	0.82	4.99	0.57	0.63	1.00	5.29	0.65
6	0.77	0.92	4.53	0.70	0.57	1.11	3.75	1.02	0.73	1.11	4.85	0.80
7	0.88	0.70	4.28	0.56	0.54	0.53	4.90	0.38	0.85	1.13	5.10	0.77
8	1.00	1.54	5.42	0.98	0.65	1.22	4.28	0.99	0.99	0.99	4.88	0.70
9	1.14	1.23	6.40	0.67	0.63	0.76	4.91	0.54	1.17	0.98	5.28	0.64
High(H)	1.37	0.97	6.97	0.48	0.62	0.63	4.63	0.47	1.47	0.86	5.40	0.55
H-L	1.38	-0.18	3.61	-0.17	-0.01	-0.44	2.37	-0.65	1.46	-0.41	3.20	-0.44
	ENet+H				GLM+H				RF			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.35	0.83	5.20	0.55	0.05	0.78	4.89	0.56	0.53	0.80	5.60	0.50
2	0.51	1.17	5.05	0.80	0.32	1.05	5.14	0.71	0.55	0.62	5.11	0.42
3	0.59	0.90	4.91	0.64	0.46	1.13	5.09	0.77	0.66	0.63	4.70	0.46
4	0.66	1.30	5.01	0.90	0.57	1.24	5.21	0.83	0.69	1.21	4.59	0.91
5	0.73	1.20	5.22	0.80	0.67	1.30	5.30	0.85	0.71	1.30	4.97	0.91
6	0.79	1.14	4.94	0.80	0.76	1.27	5.01	0.88	0.73	1.61	5.21	1.07
7	0.86	1.22	5.15	0.82	0.87	1.03	5.31	0.67	0.75	1.21	6.18	0.68
8	0.93	1.45	5.31	0.95	1.01	1.30	5.51	0.82	0.78	1.65	5.88	0.97
9	1.02	1.14	5.46	0.72	1.21	1.25	5.81	0.74	0.81	1.38	6.74	0.71
High(H)	1.15	0.93	5.67	0.57	1.57	1.38	6.69	0.71	0.88	1.59	9.29	0.59
H-L	0.80	0.11	3.90	0.09	1.52	0.60	4.02	0.51	0.35	0.79	5.79	0.47
	GBRT+H				NN1				NN2			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.30	0.69	6.09	0.39	0.02	0.85	7.54	0.39	0.36	0.85	7.78	0.38
2	0.36	0.60	5.52	0.37	0.32	1.01	6.99	0.50	0.52	1.06	6.94	0.53
3	0.51	0.56	5.38	0.36	0.48	0.75	5.93	0.44	0.60	0.97	6.22	0.54
4	0.55	0.93	4.96	0.65	0.59	1.19	5.21	0.79	0.66	0.89	5.33	0.58
5	0.59	1.23	4.87	0.88	0.68	1.19	5.04	0.82	0.70	0.96	5.02	0.66
6	0.63	1.27	4.78	0.92	0.77	1.16	4.81	0.84	0.75	1.31	4.83	0.94
7	0.67	1.23	4.78	0.89	0.86	1.19	4.95	0.83	0.79	1.15	4.55	0.88
8	0.71	1.52	5.25	1.00	0.95	1.07	4.78	0.78	0.83	1.19	4.96	0.83
9	0.74	1.29	5.54	0.81	1.07	1.26	4.99	0.88	0.88	1.09	5.17	0.73
High(H)	0.83	1.67	7.76	0.74	1.28	1.17	6.79	0.60	0.99	0.80	5.64	0.49
H-L	0.53	0.97	4.35	0.77	1.26	0.32	4.84	0.23	0.63	-0.06	4.97	-0.04
	NN3				NN4				NN5			
	Pred	Avg	Std	SR	Pred	Avg	Std	SR	Pred	Avg	Std	SR
Low(L)	0.56	1.06	7.38	0.50	0.63	0.87	7.70	0.39	0.70	0.62	6.71	0.32
2	0.64	0.91	6.96	0.45	0.69	1.12	6.71	0.58	0.72	0.73	6.07	0.42
3	0.68	0.90	5.99	0.52	0.71	1.03	6.08	0.58	0.73	1.06	5.62	0.65
4	0.70	0.81	5.65	0.50	0.73	0.71	5.63	0.44	0.74	1.05	4.99	0.73
5	0.72	1.21	4.69	0.89	0.75	0.99	5.27	0.65	0.75	1.06	4.69	0.78
6	0.75	1.31	5.11	0.89	0.76	1.34	5.31	0.87	0.75	1.31	4.78	0.95
7	0.77	1.11	4.91	0.78	0.77	1.21	4.78	0.88	0.75	1.25	4.61	0.94
8	0.79	1.10	4.94	0.77	0.78	1.22	4.88	0.86	0.76	1.07	5.15	0.72
9	0.82	1.05	5.13	0.71	0.80	0.97	5.03	0.67	0.77	1.18	5.88	0.70
High(H)	0.87	0.72	6.23	0.40	0.84	0.91	5.71	0.55	0.79	0.96	7.50	0.44
H-L	0.32	-0.35	5.26	-0.23	0.21	0.05	4.94	0.03	0.09	0.34	4.10	0.29

Note: In this table, we report the performance of prediction-sorted portfolios over the 4-year out-of-sample testing period (201701-202012). All portfolios are value weighted.

CHAPTER 2
AUTOENCODER ASSET PRICING MODELS

Research in collaboration with Bryan Kelly and Dacheng Xiu. A revised version of this paper is published on Journal of Econometrics.

We propose a new latent factor conditional asset pricing model. Like Kelly, Pruitt, and Su (KPS, 2019), our model allows for latent factors and factor exposures that depend on covariates such as asset characteristics. But, unlike the linearity assumption of KPS, we model factor exposures as a flexible nonlinear function of covariates. Our model retrofits the workhorse unsupervised dimension reduction device from the machine learning literature—autoencoder neural networks—to incorporate information from covariates along with returns themselves. This delivers estimates of *nonlinear conditional* exposures and the associated latent factors. Furthermore, our machine learning framework imposes the economic restriction of no-arbitrage. Our autoencoder asset pricing model delivers out-of-sample pricing errors that are far smaller (and generally insignificant) compared to other leading factor models.

Key words: Stock returns, conditional asset pricing model, nonlinear factor model, machine learning, autoencoder, neural networks, big data

2.1 Introduction

A recent asset pricing literature has emerged challenging the "anomaly" view of characteristic-based asset return prediction. The anomaly view suggests that certain asset attributes have the power to forecast returns above and beyond the expected return variation warranted as compensation for aggregate risk exposures. Kelly, Pruitt, and Su (KPS, 2019) provide empirical evidence that these so-called anomaly asset characteristics in fact proxy for unobservable and time-varying exposures to risk factors, and shows that characteristics contain little (if any) anomalous return predictability once their explanatory power for factor exposures has been accounted for. In other words, characteristics appear to predict returns because they help pinpoint compensated aggregate risk exposures.

The asset pricing model proposed by KPS assumes that individual returns $r_{i,t}$ possess a K-factor structure:

$$r_{i,t} = \beta(z_{i,t-1})' f_t + u_{i,t}. \tag{2.1}$$

The factors f_t are treated as latent, and the $K \times 1$ conditional factor exposure $\beta(z_{i,t-1})$ is a function of an $P \times 1$ vector of asset characteristics $z_{i,t-1}$, where P is potentially high dimensional and strictly greater than K. KPS make the simplifying assumption that the map from P characteristics to K betas is linear:

$$\beta(z_{i,t-1})' = z'_{i,t-1}\Gamma, \tag{2.2}$$

which leads to an especially tractable estimation strategy for both the beta function and the latent factors.

There are, nonetheless, no obvious theoretical or intuitive justifications for this convenient linearity assumption. To the contrary, there are many reasons to expect that this assumption is violated. Essentially all leading theoretical asset pricing models predict nonlinearities in return dynamics as a function of state variables; Campbell and Cochrane [1999], Bansal and Yaron [2004], and He and Krishnamurthy [2013] are prominent examples. Theory also predicts complex dynamics in factor risk exposures, as shown for example in the general equilibrium model of Santos and Veronesi [2004]. Moreover, Pohl et al. [2018] show that linear approximations to nonlinear models can lead to considerable errors in the model predictions for the magnitude of the equity premium or return predictability.

We generalize the factor model in (2.1) using models from the autoencoder family. Autoencoders are workhorse dimension reduction models in the field of machine learning. They can be thought of as a nonlinear, neural network counterpart to principal components analysis (PCA). Both autoencoders and PCA are unsupervised methods—they attempt to model the full panel of asset returns using only the returns themselves as inputs. The statistical content

of both methods is a bottleneck that enforces a parsimonious representation of the return data set. The PCA bottleneck uses a linear mapping from N individual returns into $K \ll N$ factors, while autoencoders allow for a nonlinear mapping through neural networks.

Neither method, in their standard form, uses information in covariates to guide dimension reduction. KPS propose "instrumented" PCA (IPCA), which allows the information in covariates to guide the reduction via equation (2.2) but remains reliant on the linear model formulation.

In this paper, we introduce a new conditional autoencoder model for individual stock returns which, like IPCA, allows covariates to help guide dimension reduction. Our autoencoder uses a neural network-based compression of returns into a low-dimensional set of factors, allowing stock characteristic covariates to have nonlinear and interactive effects on factor exposures. At the same time, we make economically guided choices in structuring the autoencoders, imposing that the factors are interpretable as portfolios, i.e., that they are linear combinations of individual equity returns. Ultimately, ours is a nonlinear conditional asset pricing model, where the nonlinearities manifest through a flexible neural network mapping of covariates into betas.

Our empirical analysis of a 60-year history of individual equity returns in the US shows that our autoencoder model dominates observable factor models in the tradition of Fama and French [1993] that use static factor betas, as well as the more sophisticated models such as the linear conditional beta specification of KPS. We follow KPS and compare models based on two statistical criteria. The first is a model's "total R^2," which describes the fraction of variation in the full panel of returns explained by contemporaneous factor realizations. This measures the model's ability to describe the riskiness, or joint return covariation, in the set of assets. The second criterion is a model's predictive R^2, which describes the fraction of return variation described by lagged conditional betas. This measures a model's ability to describe differences in risk compensation across assets.

We conduct all of our comparisons on a purely out-of-sample basis. Focusing, for example, on three-factor specifications, we find that monthly total R^2 from our preferred autoencoder, from IPCA, and from the Fama-French three-factor model are 12.6%, 13.3%, and 3.4%, respectively, while the predictive R^2 is 0.50% and 0.23% for the autoencoder and IPCA, and is negative for the Fama-French model.

We also compare the relative performance of models in economic terms. In particular, we form long-short decile spread portfolios directly sorted on out-of-sample stock return predictions from each model. Portfolios based on the three-factor autoencoder, IPCA, and Fama-French models earn annualized Sharpe Ratios of 2.16, 1.26, and -0.40, respectively, when portfolios are equal weighted. For value weighted portfolios, the respective Sharpe ratios are

0.92, 0.59, and -0.69. In summary, our conditional autoencoder model outperforms its leading competitors by a wide margin.

We contribute to a burgeoning literature using high dimensional statistical methods, including machine learning techniques, to analyze the cross section of risk and return in financial markets. The leading example in this vein is Gu et al. [2019a], who conduct a comparison of machine learning methods for predicting the panel of individual US stock returns. That paper outlines a new research agenda marrying machine learning with the study of asset risk premia.[1] Gu et al. [2019a] focus on supervised prediction models but take no stand on the risk-return tradeoff. In other words, their approaches do not constitute asset pricing models. In this paper, we develop asset pricing models using unsupervised and semi-supervised learning methods that model the risk-return tradeoff explicitly. Autoencoders are critical tools in the machine learning suite that have enjoyed success in wide range of practical applications.[2] Our contribution links the methodology literature on autoencoders with the large finance literature on factor pricing models.

KPS, who focus specifically on conditional factor pricing models, is the closest predecessor to our analysis.[3] They unify the literature on linear latent factor APT models [starting from Ross, 1976] with the literature on characteristic-based "anomaly" return prediction. One of our contributions is to extend their work by allowing for a general nonlinear specification of the return factor structure. To do so, we augment the traditional autoencoder by embedding a neural network in the specification of conditional betas, which allows characteristics to determine risk exposures through flexible nonlinearities and interactions, generalizing the linear "instrumented" beta specification of KPS.[4]

Another related predecessor is Kozak et al. [2018], who propose an approach to factor analysis for asset pricing using principal components from "anomaly"-sorted portfolios [reviving an earlier literature on this approach exemplified by Chamberlain and Rothschild, 1983, Connor and Korajczyk, 1986]. This approach, however, fails to explain the risk-return trade-

1. Other related asset pricing papers include Feng et al. [2019b], Freyberger et al. [2019], Kozak et al. [2019], Feng et al. [2019a], Giglio and Xiu [2018], and Gagliardini et al. [2016], among others.

2. See, for example, Gallinari et al. [1987], Bourlard and Kamp [1988], Hinton and Zemel [1994], and Goodfellow et al. [2016].

3. A related contemporaneous paper that builds on the instrumented beta approach of KPS using kernel methods is Kozak [2019].

4. Our approach to autoencoder learning for factor models is also related to work relying on kernel or sieve approximations, e.g., Connor et al. [2012] and Fan et al. [2016]. Recent work also considers machine learning tools for factor model estimation, such as nuclear norm penalization [Bai and Ng, 2017, Moon and Weidner, 2018] and partial least squares [Kelly and Pruitt, 2015]. Our autoencoder approach allows for flexible (and conditional) model structures, and uses stochastic gradient descent to manage the computational complexity in the resulting big data and high parameter setting. That said, a thorough theoretical analysis on the statistical properties of this alternative learning method is left for future research.

off when the test assets are individual stocks. Our conditional autoencoder model does not suffer this shortcoming. It accurately describes the risk and expected return successfully for individual stocks as well as for anomaly or other stock portfolios. More broadly, we show that our conditional autoencoder formulation is a valid asset pricing model. It is equivalent to a nonparametric model for a stochastic discount factor, and imposes the economic restriction of no-arbitrage pricing. One illustration of the conditional autoencoder's empirical success as a pricing model is that it correctly prices so-called anomaly portfolios with small and insignificant pricing errors. The pricing errors in our model, which are measured on a purely out-of-sample basis, are a fraction of the magnitude of those from traditional Fama-French factor models.

The rest of the paper is organized as follows. In Section 2.2, we set up the model and present our methodology. Section 2.3 presents our empirical studies. Section 2.4 provides Monte Carlo simulations that demonstrate the performance of our procedures. Section 2.5 concludes. The appendix contains mathematical proofs and detailed algorithms.

2.2 Methodology

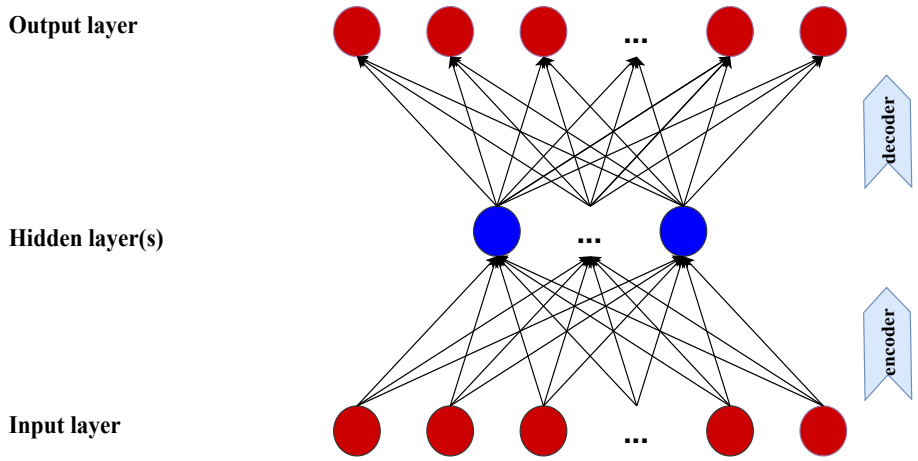
In this section we first describe standard autoencoders, then we introduce our new conditional autoencoder that leverages covariates as conditioning information. We highlight parallels between autoencoder models and widely studied factor pricing models, and illustrate that linear factor models (and their associated PCA and IPCA estimators) are a special case of autoencoders. Finally, we describe our autoencoder estimation approach.

2.2.1 Standard Autoencoder

An autoencoder is a special neural network in which the outputs attempt to approximate the input variables. The input variables pass through a small number of neurons in the hidden layer(s), forging a compressed representation of the input (encoding), which is then unpacked and mapped to the output layer (decoding). Because no other variables are used in this model besides the inputs, an autoencoder is an unsupervised learning device.

Neural network models (including autoencoders) with L hidden layers can be written using the following recursive 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 $r_k^{(l)}$, and the vector of all outputs for this layer as $r^{(l)} = (r_1^{(l)}, \ldots, r_{K^{(l)}}^{(l)})'$. In each hidden layer, inputs from the previous layer are transformed according to a nonlinear activation function $g(\cdot)$ before being passed to the next layer. To initialize the network, the input layer uses the cross section of returns as raw predictors, $r^{(0)} = r = (r_1, \ldots, r_N)'$. The recursive output formula for the neural network in

Figure 2.1: Standard Autoencoder Model



Note: This figure describes a standard autoencoder with one hidden layer. The output and input layers are identical, while the hidden layer is a low dimensional compression of inputs variables into latent factors, which can be expressed as weighted linear combinations of input variables.

layer $l > 0$ is then

$$r^{(l)} = g\left(b^{(l-1)} + W^{(l-1)}r^{(l-1)}\right), \tag{2.3}$$

where $W^{(l-1)}$ is a $K^{(l)} \times K^{(l-1)}$ matrix of weight parameters, and $b^{(l-1)}$ is a $K^{(l)} \times 1$ vector of so-called bias parameters. Throughout this paper we use as our nonlinear activation function the rectified linear unit (ReLU), $g(y) = \max(y, 0)$.[5] The number of parameters in each hidden layer l is $K^{(l)}(1 + K^{(l-1)})$. The final output of the autoencoder,

$$G(r, b, W) = b^{(L)} + W^{(L)}r^{(L)}, \tag{2.4}$$

which shares the same dimension as the input, is employed to approximate r itself. Figure 2.1 illustrates the architecture of a simple linear autoencoder with a single hidden layer.

Static Linear Factor Models as a Special Case

The autoencoder is a dimension reduction device. Its objective is to learn a low dimensional representation of the inputs by passing them through a central hidden layer with many fewer neurons than the dimension of r. It thus shares the same spirit as PCA, but is more flexible in that it allows for nonlinear compression of the inputs. In this section we explore the connection

5. Without ambiguity, we regard $g(y)$ as an entry-wise vector-valued function whose length is equal to that of its input vector y.

between autoencoders and PCA.

In the finance literature, some of the most commonly studied models of asset returns assume a linear latent factor specification with static loadings:[6]

$$r_t = \beta f_t + u_t, \tag{2.5}$$

where r_t is a vector of returns in excess of the risk free rate, f_t is a $K \times 1$ vector of factor returns, u_t is a $N \times 1$ vector of idiosyncratic errors (uncorrelated with f_t), and β is an $N \times K$ matrix of factor loadings. This resembles that standard factor model setting whose econometric properties are studied in Bai and Ng [2002], Bai [2003], and Giglio and Xiu [2018], among others.

Stacking the time series vectors, the matrix form of the factor model is

$$R = \beta F + U.$$

Following Stock and Watson [2002a] and Bai and Ng [2002], a factor model can be estimated with PCA on the covariance matrix of returns.[7] Equivalently, a singular value decomposition (SVD) of $\bar{R}$, the demeaned returns of R, yields estimates of factors and factor loadings directly.[8] That is,

$$\bar{R} = \widehat{P}\Lambda\widehat{Q} + \widehat{U}, \tag{2.6}$$

where $\widehat{P}$ and $\widehat{Q}$ are, respectively, the $N \times K$ and $K \times T$ matrices of left and right singular vectors, and $\widehat{U}$ is a $N \times T$ matrix of residuals.

The machine learning literature has long recognized the close connection between autoencoders and PCA [e.g., Baldi and Hornik, 1989] and, by extension, between autoencoders and latent factor asset pricing models. When the autoencoder has one hidder layer and a linear activation function, it is equivalent to the PCA estimator for linear factor models described above.

6. For example, Connor and Korajczyk [1986] and Kozak et al. [2019].

7. A subtle consideration in estimating asset pricing models with PCA is choosing whether to impose the zero intercept no-arbitrage restriction. Imposing the restriction amounts to applying PCA to the uncentered second moment matrix of excess returns, rather than to the (centered) covariance matrix.

8. This approach is applicable with any (fixed) number of factors K. An extensive literature studies methods for choosing K for a large N and large T panel, including Bai and Ng [2002], Hallin and Liška [2007], Amengual and Watson [2007], Alessi et al. [2010], Kapetanios [2010], Onatski [2010], Ahn and Horenstein [2013], and Aït-Sahalia and Xiu [2017]. Throughout the paper, we will treat K as a tuning parameter, which in principle can be estimated via cross validation.

More specifically, we can write the one-layer, linear autoencoder with K neurons as:

$$r_t = b^{(1)} + W^{(1)}(b^{(0)} + W^{(0)} r_t) + u_t, \tag{2.7}$$

where $W^{(0)}$, $W^{(1)}$, $b^{(1)}$ and $b^{(0)}$ are $K \times N$, $N \times K$, $N \times 1$, and $K \times 1$ matrices of parameters, respectively. The model can be estimated by solving the following optimization problem:

$$\begin{aligned} &\min_{b,W} \sum_{t=1}^{T} \left\| r_t - \left(b^{(1)} + W^{(1)}(b^{(0)} + W^{(0)} r_t) \right) \right\|^2 \\ =&\min_{b,W} \left\| R - \left(b^{(1)} \iota' + W^{(1)}(b^{(0)} \iota' + W^{(0)} R) \right) \right\|_{\mathrm{F}}^2, \end{aligned} \tag{2.8}$$

where the F subscript denotes the Frobenius norm, and ι is $T \times 1$ vector of 1s. The next proposition establishes the link between this simple autoencoder and the PCA estimator.

Proposition 1. *The optimal solution to (2.8) is given by,*

$$\widehat{W}^{(1)} = \widehat{P} A, \quad \widehat{W}^{(0)} = (\widehat{W}^{(1)\prime} \widehat{W}^{(1)})^{-1} \widehat{W}^{(1)\prime}, \quad \widehat{b}^{(1)} = \bar{r} - \widehat{W}^{(1)} \widehat{b}^{(0)} - \widehat{W}^{(1)} \widehat{W}^{(0)} \bar{r}, \quad \widehat{b}^{(0)} = a,$$

where A is any $K \times K$ non-singular matrix, a is a constant scalar, $\bar{r}$ is the sample average of r_t, and $\widehat{P}$ is from equation (2.6).[9]

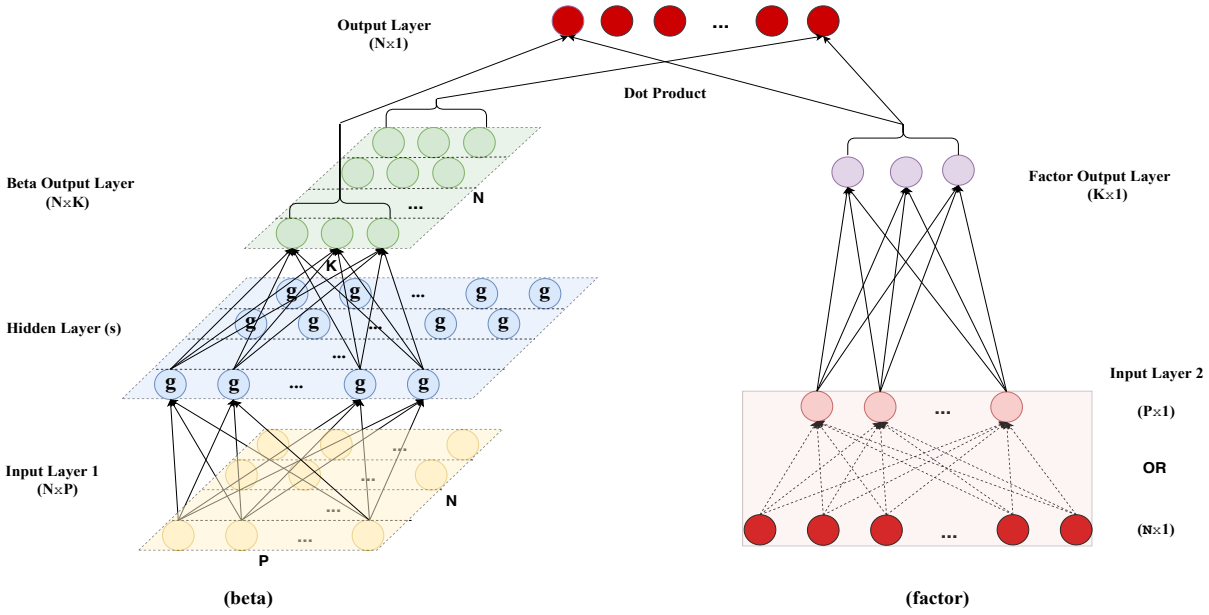
Proposition 1 shows that the linear autoencoder with a hidden layer of K neurons is a linear factor model with K latent factors. The estimated factor loadings are $\widehat{W}^{(1)}$, and the estimated factors are $\widehat{W}^{(0)} R$. These span the same spaces as $\widehat{P}$ and $\widehat{Q}$ in equation (2.6). Needless to say, autoencoder models are more general than the linear factor model as they allow for dimension reduction via layers of nonlinear transformations of r_t. This additional flexibility has proven valuable in a variety of applications outside of finance. For example, Hinton and Salakhutdinov [2006] show that deep autoencoders handily outperform shallow or linear autoencoders for image recognition.

2.2.2 Extending the Autoencoder Model to Include Covariates

The static linear factor model has been an extremely productive tool for studying asset returns, but recent research highlights a number of its limitations. The distribution of asset returns is well known to be highly time-varying, and static factor models abstract from a wealth of relevant conditioning information. For example, KPS demonstrate large empirical gains from

9. We provide a proof for the equivalence between the standard (centered) PCA and the linear autoencoder (with biases). The equivalence in the uncentered case is similar.

Figure 2.2: Conditional Autoencoder Model



Note: This figure presents the diagram of an autoencoder augmented to incorporate covariates in the factor loading specification. The left-hand side describes how factor loadings β_{t-1} at time $t-1$ (in green) depend on firm characteristics Z_{t-1} (in yellow) of the input layer 1 through an activation function g on neurons of the hidden layer. Each row of yellow neurons represents the $P \times 1$ vector of characteristics of one ticker. The right-hand side describes the corresponding factors at time t. f_t nodes (in purple) are weighted combinations of neurons of the input layer 2, which can either be P characteristic-managed portfolios x_t (in pink) or N individual asset returns r_t (in red). In the latter case, the input layer 2 is exactly what the output layer aims to approximate, which is the same as a standard autoencoder.

incorporating asset-specific covariates in the specification of factor loadings. Not only do these covariates improve the estimation of loadings, they also indirectly improve estimates of the latent factors themselves. Their model formulation amounts to a combination of two linear models: One linear specification for the latent factors, shown in equation (2.1), and another linear specification for conditional betas, shown in (2.2).

While the standard autoencoder in (2.5) is a powerful tool for dimension reduction, it shares the same limitation as PCA that it does not leverage conditioning variables to identify the factor structure, and instead relies only on returns themselves. To overcome this limitation, we design a new neural network structure by augmenting the standard autoencoder model to incorporate covariates.

Figure 2.2 illustrates the basic structure of our conditional autoencoder. The left side of the network models factor loadings as a nonlinear function of covariates (e.g., asset characteristics), while the right side network models factors as portfolios of individual stock returns. At the

highest level, the mathematical representation of the model is identical to equation (2.1):

$$r_{i,t} = \beta'_{i,t-1} f_t + u_{i,t}. \tag{2.9}$$

The first key difference between our model and IPCA is in the formulation of conditional betas. We specify the $K \times 1$ vector $\beta_{i,t-1}$ as a neural network model of lagged firm characteristics, $z_{i,t-1}$. The recursive formulation for the nonlinear beta function is:

$$z^{(0)}_{i,t-1} = z_{i,t-1}, \tag{2.10}$$

$$z^{(l)}_{i,t-1} = g\left(b^{(l-1)} + W^{(l-1)} z^{(l-1)}_{i,t-1}\right), \quad l = 1, ..., L_\beta, \tag{2.11}$$

$$\beta_{i,t-1} = b^{(L_\beta)} + W^{(L_\beta)} z^{(L_\beta)}_{i,t-1}. \tag{2.12}$$

Equation (2.10) initializes the network as a function of the baseline characteristic data, $z_{i,t-1}$. The equations in (2.11) describe the nonlinear (and interactive) transformation of characteristics as they propagate through hidden layer neurons.[10] Equation (2.12) describes how a set of K-dimensional factor betas emerge from the terminal output layer. This formalizes the left side of Figure 2.2.

On the right side of Figure 2.2, we see an otherwise standard autoencoder for the factor specification. The recursive mathematical formulation of the factors is:

$$r^{(0)}_t = r_t, \tag{2.13}$$

$$r^{(l)}_t = \widetilde{g}\left(\widetilde{b}^{(l-1)} + \widetilde{W}^{(l-1)} r^{(l-1)}_t\right), \quad l = 1, ..., L_f, \tag{2.14}$$

$$f_t = \widetilde{b}^{(L_f)} + \widetilde{W}^{(L_f)} r^{(L_f)}_t. \tag{2.15}$$

Equation (2.13) initializes the network with the vector of individual asset returns, r_t. Equations in (2.14) transform and compress the dimensionality of returns as they propagate through hidden layers. Equation (2.15) describes the final set of K factors at the output layer. Throughout our empirical analysis, we assume a single linear layer on the factor network, that is, $L_f = 1$, in that this structure maintains the economic interpretation of factors: they are themselves portfolios (linear combination of underlying asset returns).

At last, the "dotted operation" multiplies the $N \times K$ matrix output from the beta network with the $K \times 1$ output from the factor network to produce the final model fit for each individual

10. Equations (2.11) and (2.14) assume the convention that $z^{(l)}_{i,t-1}$ and $r^{(l)}_t$ are vectors that stack the output from all neurons in the l^{th} layer.

asset return.

In practice, using the full cross section of individual stock returns in the factor network faces two daunting obstacles. The first is that the number of individual firms in our sample is roughly 30,000, which means that the number of weight parameters in the factor network can be astronomical, while the number of time series observations in our data set is a mere 720. Second, the panel is extremely unbalanced—in any given month, we have on average around 6,000 non-missing stocks, thus most of the stock-level weight parameters would require estimation from very few time series observations.

We therefore make one key modification to the factor side of the model that massively reduces the model's computational cost up front. Instead of initializing the network with the full cross section of stock returns in equation (2.13), we instead initialize it with a set of portfolios, defined as

$$x_t = (Z'_{t-1}Z_{t-1})^{-1}Z'_{t-1}r_t. \tag{2.16}$$

This $P \times 1$ vector is a set of portfolios that are dynamically re-weighted (or "managed") on the basis of stock-level characteristics. The j^{th} element of x_t is akin to a return on a long-short portfolio constructed by sorting stocks based on the j^{th} characteristic. Initializing the model with $r_t^{(0)} = x_t$ accomplishes three things at once. First, it performs a preliminary reduction of the data that eliminates tens of thousands of parameters from the model. This preprocessing step in (2.16) can be viewed as adding a new initial layer to the factor neural network that dynamically (as a function of Z_{t-1}) collapses the N returns, r_t, down to P neurons, x_t, before proceeding with the rest of the network propagation. Second, it sidesteps issues of panel incompleteness, since portfolios are formed from the subset of stocks that are non-missing at each point in time. Third, it connects the factor autoencoder to the finance literature on characteristic-managed portfolios, including KPS, Feng et al. [2019a], Kozak et al. [2019], and Giglio and Xiu [2018]. KPS and Giglio and Xiu [2018], in particular, show that conditional linear factor models for individual stocks can be recast as static factor analysis on characteristic-managed portfolios.

Conditional Linear Factor Models as a Special Case

Just like the autoencoder model nests the static linear factor model, the augmented autoencoder nests the IPCA factor model proposed by KPS as a special case. KPS propose a method called IPCA to estimate their linear, time-varying beta model. IPCA solves the optimization problem:

$$\min_{\Gamma,F} \sum_{t=1}^{T}\sum_{i=1}^{N} \left\| r_{i,t} - z'_{i,t-1}\Gamma' f_t \right\|^2 = \min_{\Gamma,F} \sum_{t=1}^{T} \left\| r_t - Z_{t-1}\Gamma' f_t \right\|^2. \tag{2.17}$$

where $F = (f_1, f_2, \ldots, f_T)$ and $Z_t = (z'_{1,t}, z'_{2,t}, \ldots, z'_{N,t})'$, and subject to restrictions that identify a unique rotation of factors.[11]

For comparison, consider a particularly simple version of the conditional autoencoder that uses a linear activation function and one layer of K neurons on both the beta side and the factor side of the network. In this case, $\beta'_{i,t} = Z_{t-1}W'_0$ and $f_t = W_1 x_t$, so the the estimation objective of the conditional autoencoder is:[12]

$$\min_{W_0, W_1} \sum_{t=1}^{T} \left\| r_t - Z_{t-1} W'_0 W_1 x_t \right\|^2 . \tag{2.18}$$

The next proposition formalizes the equivalence between IPCA and this linear conditional autoencoder.

Proposition 2. *The solution to* (2.18) *is equivalent to the solution of* (2.17) *if* $Z'_t Z_t = \Sigma$ *for a constant matrix* Σ.

In the general case where $Z'_t Z_t$ is non-constant, the two estimators are similar but no longer equivalent (as we can see from the proof). We find that the empirical performance of (2.17) and (2.18) is similar in our data.

2.2.3 Regularized Autoencoder Learning

Autoencoders, like neural networks more broadly, have many advantages relative to traditional linear factor models. In particular, the high capacity of a neural network model enhances its flexibility to construct the most informative features from data. With enhanced flexibility, however, comes a higher propensity to overfit. Following Gu et al. [2019a], we take a variety of measures to alleviate overfitting, including a careful design of the empirical strategy and the extensive use of regularization.

Training, Validation, and Testing

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

11. The identifying assumptions are

$$\Gamma\Gamma' = \mathbb{I}_K, \quad FF' \text{ is a diagonal matrix with descending diagonal entries}, \quad F\iota \geq 0.$$

These restrictions place no economic restrictions on the model and solely serve to pin down a uniquely identified solution to the first-order conditions.

12. Without loss of generality, we suppress the bias term, i.e., b_0, in the neural networks, as it can instead be represented via a constant in the list of conditioning variables.

specific set of tuning hyperparameter values. These are critical to the performance of machine learning methods as they control model complexity.

The second, or "validation," sample is used for tuning the hyperparameters. We construct fitted values for data points in the validation sample based on the estimated model from the training sample. Next, we calculate the objective function based on errors from the validation sample, and iteratively search for hyperparameters that optimize the validation objective.

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 out-of-sample performance.

Regularization Techniques

The most common machine learning device for guarding against overfitting is to append a penalty to the objective function in order to favor more parsimonious specifications. This "regularization" approach mechanically deteriorates a model's in-sample performance in the hope of improving its stability out-of-sample. This will be the case when penalization manages to reduce the model's fit of noise while preserving its fit of the signal.

Let $\mathcal{L}(\theta; \cdot)$ denote the objective function to optimize the autoencoder (2.9), where θ summarizes the weight parameters in the loading and factor networks of (2.10) through (2.15). We define our estimation objective to be

$$\mathcal{L}(\theta; \cdot) = \frac{1}{NT} \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| r_{i,t} - \beta'_{i,t-1} f_t \right\|^2 + \phi(\theta; \cdot), \tag{2.19}$$

where $\phi(\theta)$ is a penalty function that regularizes the model. There are many choices for the penalty function $\phi(\cdot)$; we use LASSO, or "l_1," penalization, which takes the form

$$\phi(\theta; \lambda) = \lambda \sum_{j} |\theta_j|.$$

The fortunate geometry of the LASSO penalty sets coefficients on a subset of covariates to exactly zero. In this sense, the LASSO imposes sparsity on weight parameters, encouraging insignificant weights to vanish. The LASSO penalty involves a non-negative hyperparameter,

λ, which is determined in the validation sample.

In addition to l_1-penalization, we employ a second machine learning regularization tool known as "early stopping." It begins from an initial parameter guess that imposes parsimonious parameterization (for example, setting all θ values close to zero). In each step of the optimization algorithm, the parameter guesses are gradually updated to reduce fitting errors in the training sample. At each new guess, estimates 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 fitting errors are minimized in the training sample, hence its name (see Algorithm 6). By ending the parameter search early, parameters are shrunken toward the initial guess, and this is how early stopping regularizes against overfit. It is a popular substitute to "l_2"-penalization of θ parameters because it achieves regularization at a much lower computational cost.

As a third regularization technique, we adopt an ensemble approach in training our neural networks. In particular, we use multiple random seeds, say, 10, to initialize neural network estimation and construct model predictions by averaging estimates from all networks. This enhances the stability of the results because the stochastic nature of the optimization can cause different seeds to settle at different optima.

Optimization Algorithms

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 gradient 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 at each iteration. This approximation sacrifices accuracy for enormous acceleration of the optimization routine. 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 adaptive moment estimation algorithm (Adam, Algorithm 5), an efficient version of SGD introduced by Kingma and Ba [2014] that computes adaptive learning rates for individual parameters using estimates of first and second moments of the gradients.

We also adopt "batch normalization" (Ioffe and Szegedy [2015], Algorithm 7), a simple technique for controlling the variability of predictors across different regions of the network and across different datasets. 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 layer in each training step (a "batch"), the algorithm cross-sectionally de-means and variance standardizes the batch inputs to restore the representation power of the unit.

2.3 An Empirical Study of US Equity

2.3.1 Data

We analyze the same dataset studied in Gu et al. [2019a], which contains monthly individual stock returns from the Center for Research in Securities Prices (CRSP) for all firms listed in the three major exchanges: NYSE, AMEX, and NASDAQ. We use the Treasury bill rate to proxy for the risk-free rate from which we calculate individual excess returns. Our sample begins in March 1957 (the start date of the S&P 500) and ends in December 2016, totaling 60 years.

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 (61 of which are updated annually, 13 updated quarterly, and 20 updated monthly), see Gu et al. [2019a] for a full list. Most of these characteristics are released to the public with a delay. To avoid a forward-looking bias, we assume that monthly characteristics are delayed by at most one month, quarterly releases are delayed with a four month lag, and annual releases with a six month lag. Thus, we match realized returns at month t with the most recent monthly characteristics at the end of month $t-1$, the most recent quarterly data as of $t-4$, and most recent annual data as of $t-6$. Observations are occasionally missing some characteristics. We replace a missing characteristic with the cross-sectional median of that characteristic during that month.

Distributions of some characteristics are highly skewed and leptokurtic. Following a common tack in the literature that avoids undue influence of outlying observations, we rank-normalize all characteristics into the interval $(-1, 1)$ for each month t. We then form 94 managed portfolios using (2.16). We also include one equal-weighted market portfolio that corresponds to a constant regressor in Z_{t-1}.

Unlike the existing literature, we do not impose any filters based on stock prices or share codes, or rule out financial firms. Past literature has imposed these filters in large part because they find it difficult to reconcile the return behavior of low price stocks, uncommon share codes, and financial sector stocks with the rest of the sample. We have no such difficulty thanks to the superior capacity of our model and the rich feature sets we allow for in our framework.

The total number of stocks in our sample is nearly 30,000, with the average number of stocks per month exceeding 6,200.

2.3.2 Models Comparison Set

We compare a range of latent factor models in our empirical analysis. The first model, which we refer to as "PCA," corresponds to specification (2.5). It assumes a linear functional form, constant betas, no conditioning information, and is estimated via PCA.[13] The second model we refer to as "IPCA" and follows the KPS specification of (2.1) and (2.2). In particular, it assumes a linear factor structure and conditional betas that are likewise linear in covariates.

We then consider a range of conditional autoencoder (CA) architectures with varying degrees of complexity. The simplest, which we denote CA_0, uses a single linear layer in both the beta and factor networks as described in (2.18), making it similar (but not identical) to IPCA. Next, CA_1 adds a hidden layer with 32 neurons in the beta network. Finally, CA_2 and CA_3 add a second and third hidden layer, with 16 and 8 neurons respectively, to the beta side.

CA_0 through CA_3 all maintain a one-layer linear specification on the factor side of the model. In these cases, the only variation in factor specification is in the number of neurons, which we allow to range from 1 to 6, and which corresponds to the number of factors in the model.

We also compare the autoencoder specifications against benchmark models with observable factors. We refer to these models collectively as "FF," which possess 1 to 6 factors. The first observable factor is the excess market return, then we add SMB, HML, and UMD, sequentially. The five-factor model is the market, SMB, HML, CMA, and RMW, and the six-factor model again appends UMD.[14]

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 one year. We maintain the same size of the validation sample, but roll it forward to include the most recent twelve months.

13. The universe of individual stocks changes over time, so our PCA estimation must cope with unbalanced panels. We use an EM algorithm for PCA [Stock and Watson, 2002a]. The IPCA algorithm devised by KPS is robust to missing data. Likewise, the SGD algorithm for autoencoder models is not affected by missing data, because individual stock returns across periods are collected in a single pool from which random batches of observations are drawn.

14. Market, SMB, HML, CMA, RMW, and UMD factor returns are from Ken French's website.

2.3.3 Statistical Performance Evaluation

We evaluate out-of-sample model performance using the total and predictive R^2s defined by KPS. These pool errors across firms and over time into grand panel-level assessments of each model. The total R^2 quantifies the explanatory power of contemporaneous factor realizations, and thus assesses the model's description of individual stock riskiness:

$$R^2_{\text{total}} = 1 - \frac{\sum_{(i,t)\in OOS}(r_{i,t} - \widehat{\beta}'_{i,t-1}\widehat{f}_t)^2}{\sum_{(i,t)\in OOS} r^2_{i,t}}. \tag{2.20}$$

The OOS set indicates that fits are only assessed on the testing subsample, whose data never enter into model estimation or tuning.

The predictive R^2 assesses the accuracy of model-based predictions of future individual excess stock returns. This quantifies a model's ability to explain panel variation in risk compensation. It is defined as

$$R^2_{\text{pred}} = 1 - \frac{\sum_{(i,t)\in OOS}(r_{i,t} - \widehat{\beta}'_{i,t-1}\widehat{\lambda}_{t-1})^2}{\sum_{(i,t)\in OOS} r^2_{i,t}}, \tag{2.21}$$

where $\widehat{\lambda}_{t-1}$ is the prevailing sample average of $\widehat{f}$ up to month $t-1$.

Table 2.1 reports the out-of-sample total R^2 for individual stocks, r_t. The best overall model in terms of explained out-of-sample return variation is IPCA with six-factors, which delivers a 14.5% total R^2. It is closely followed by the conditional autoencoder with one hidden beta layer of 32 neurons (CA_1) and six factors, which achieves an out-of-sample R^2 of 14.3%. Other CA models are similar but slightly weaker.

Interestingly, and somewhat surprisingly, the worst performing models are those with observable factors. We can trace the poor performance of observable factor models to two features of our empirical design. The first is that we infrequently re-estimate model parameters. At the individual stock level, betas are highly time varying, and infrequent re-estimation clearly reveals a shortcoming of traditional observable factors and their static beta formulations. In contrast, we find that infrequent re-estimation has little effect on the out-of-sample fits for conditional models (both IPCA and CA versions), likewise revealing the comparative strength of these methods—their ability to capture time variation through covariates rather than through ad hoc rolling parameter updates. Second, we analyze a much larger cross section of stocks (nearly 30,000) than typically studied in the literature (e.g., roughly 12,000 in KPS). This reveals that observable factors are poorly suited to describe the performance of the much larger universe of stocks that we cover. Note that, even though our current data set is much larger than that of KPS, the performance of IPCA is remarkably robust to this

change in environment.

Table 2.1 also reports the out-of-sample total R^2 at the level of managed portfolios, x_t. Given our predictions at the individual equity level, the prediction at the portfolio level is immediate available. Model refitting is not needed as the portfolio weights are known ex ante (see Gu et al. [2019a] for more discussion on this "bottom-up" approach for prediction). These portfolios are large and diversified collections of individual stocks, thus much of the idiosyncratic risk in the data is averaged out. As a result, total R^2 at the portfolio-level tends to be far higher. It is still the case that IPCA provides the best fit, followed by CA_1. The comparative performance of observable factor FF models is much improved at the portfolio level,[15] but nonetheless dominated by conditional latent factor models.

Next, Table 2.2 compares models in terms of predictive R^2. Whereas IPCA dominated in terms of total R^2, its predictive R^2 of 0.3% per month is nearly doubled by the predictive power of (deep) conditional autoencoders. CA_1, CA_2, and CA_3 generate a predictive R^2 of 0.53%, 0.58%, and 0.57%, respectively. All of conditional models, including IPCA and CA_0, dramatically outperform the static FF and PCA models, which generally fail to produce any out-of-sample predictability whatsoever.

2.3.4 Economic Performance Evaluation

It is difficult to infer the economic contribution of a model from R^2 alone. To assess model performance in economic terms, we evaluate how return predictions from each model translate into Sharpe ratios for portfolios formed based on those predictions.

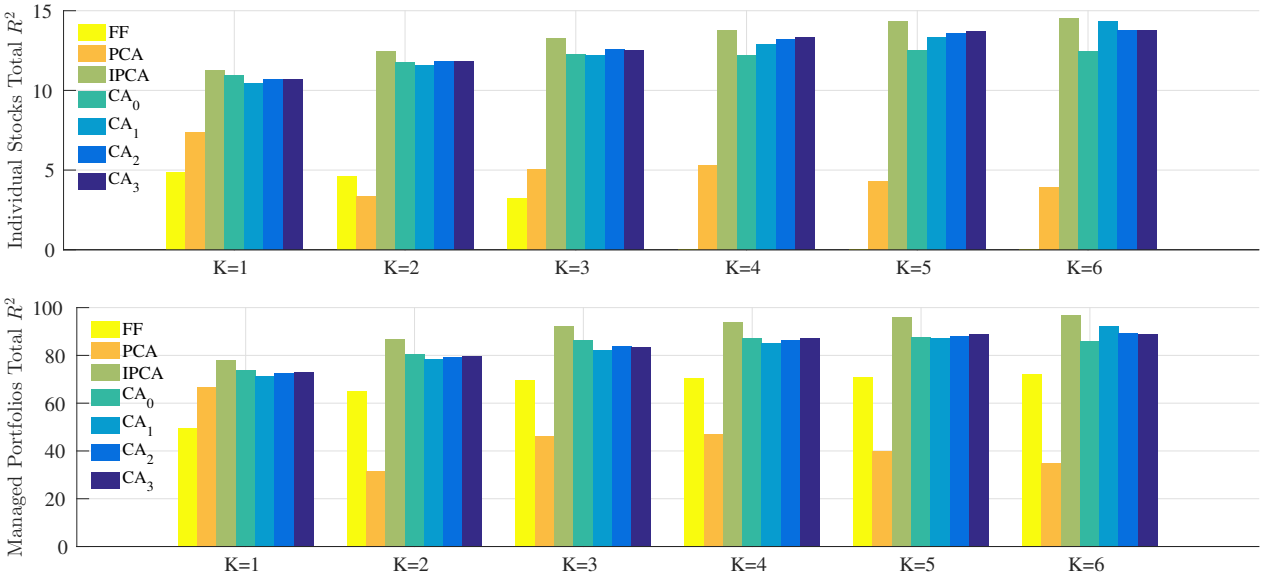
For each model, we sort stocks into deciles based on the model's out-of-sample return forecasts. We construct a zero-net-investment portfolio that buys the highest expected return stocks (decile 10) and sells the lowest (decile 1). We rebalance portfolios each month, and consider both equal-weighted and value-weighted portfolios.

Table 2.3 reports the annualized Sharpe ratios of these 10-1 spread portfolios over our 30-year out-of-sample period. The results essentially recast the model comparison of predictive R^2 in terms of economic magnitudes. The overall best performing portfolio is that based on the conditional autoencoder with two hidden beta layers, CA_2. This model achieves a Sharpe ratio of 2.63 for the equal-weighted portfolio, and 1.53 with value weights. The performance of CA_1 and CA_3 is only slightly lower. Following the nonlinear conditional autoencoders, the best model is IPCA, which delivers Sharpe ratios of 2.25 and 0.96 with equal and value weights, respectively (and which outperforms the linear conditional autoencoder CA_0). Fi-

15. Intuitively, dynamically reweighting portfolios to maintain roughly constant characteristic values reduces time variation in portfolio betas and thus gives static factor models (including PCA) a better chance to fit the data.

Table 2.1: Out-of-Sample $R^2_{\text{total}}(\%)$ Comparison

Model	Test Assets	K					
		1	2	3	4	5	6
FF	r_t	4.8	4.6	3.4	0.1	-2.3	-6.1
	x_t	49.4	64.8	69.5	70.4	71.1	72.2
PCA	r_t	7.3	3.3	5.0	5.3	4.2	3.9
	x_t	66.6	31.7	46.2	47.1	39.8	34.8
IPCA	r_t	11.2	12.4	13.3	13.7	14.3	14.5
	x_t	78.0	86.8	92.1	93.8	96.0	96.7
CA_0	r_t	10.9	11.8	12.3	12.2	12.5	12.4
	x_t	73.7	80.4	86.2	87.1	87.7	85.9
CA_1	r_t	10.4	11.5	12.2	12.9	13.4	14.3
	x_t	71.4	78.3	82.2	85.2	87.2	92.2
CA_2	r_t	10.7	11.8	12.6	13.2	13.6	13.8
	x_t	72.7	79.5	84.0	86.4	88.2	89.3
CA_3	r_t	10.7	11.8	12.5	13.3	13.7	13.8
	x_t	73.1	79.9	83.6	87.1	88.8	89.0



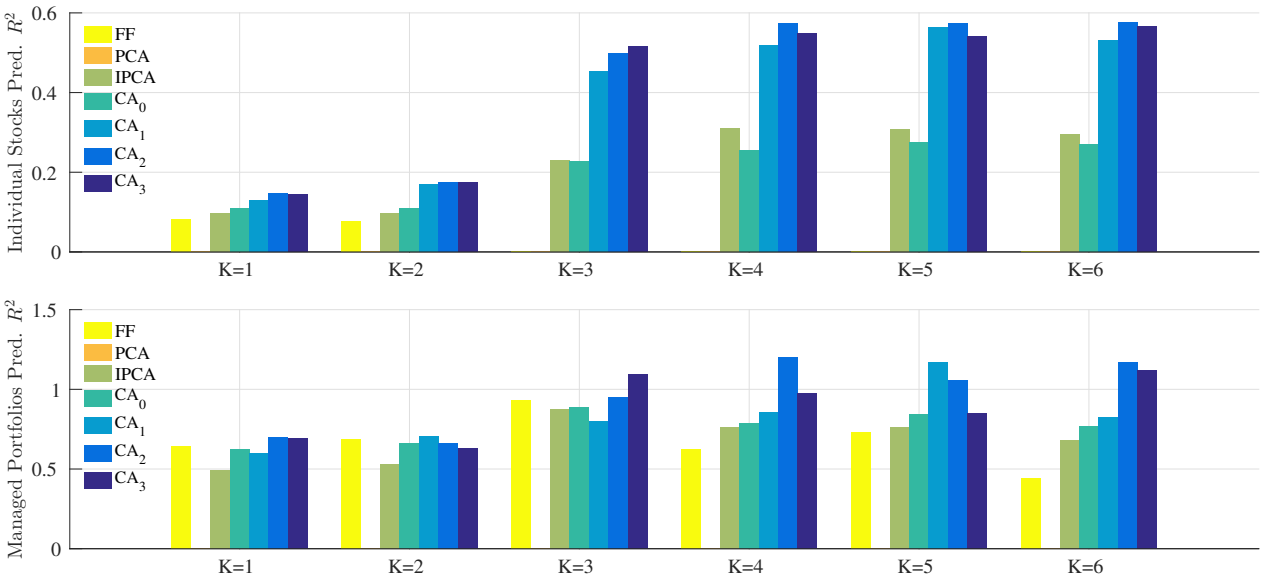
Note: In this table, we report the out-of-sample total $R^2(\%)$ for individual stocks r_t and managed portfolios x_t using observable factor models (FF), PCA, IPCA, and conditional autoencoders CA_0 through CA_3. In all cases, the number of factors K varies from 1 to 6.

nally, corroborating the R^2 results above, static linear models FF and PCA broadly exhibit poor out-of-sample portfolio performance.

To evaluate the multi-factor mean-variance efficiency of each model, we report the ex ante unconditional tangency portfolio Sharpe ratio among factor portfolios. We calculate out-of-sample factor returns following the same re-estimation approach described earlier. The

Table 2.2: Out-of-Sample $R^2_{\text{pred}}(\%)$ Comparison

Model	Test Assets	K					
		1	2	3	4	5	6
FF	r_t	0.08	0.08	< 0	< 0	< 0	< 0
	x_t	0.65	0.69	0.93	0.62	0.73	0.45
PCA	r_t	< 0	< 0	< 0	< 0	< 0	< 0
	x_t	< 0	< 0	< 0	< 0	< 0	< 0
IPCA	r_t	0.10	0.10	0.23	0.31	0.31	0.30
	x_t	0.49	0.53	0.88	0.76	0.76	0.68
CA_0	r_t	0.11	0.11	0.23	0.25	0.27	0.27
	x_t	0.63	0.66	0.89	0.79	0.84	0.77
CA_1	r_t	0.13	0.17	0.45	0.52	0.56	0.53
	x_t	0.60	0.70	0.80	0.85	1.17	0.83
CA_2	r_t	0.15	0.17	0.50	0.57	0.57	0.58
	x_t	0.70	0.66	0.95	1.20	1.06	1.17
CA_3	r_t	0.14	0.17	0.52	0.55	0.54	0.57
	x_t	0.69	0.63	1.10	0.97	0.85	1.12



Note: In this table, we report the out-of-sample predictive $R^2(\%)$ for individual stocks r_t and managed portfolios x_t using observable factor models (FF), PCA, IPCA, and conditional autoencoders CA_0 through CA_3. In all cases, the number of factors K varies from 1 to 6.

tangency portfolio return for a set of factors is constructed on a purely out-of-sample basis by using the mean and covariance matrix of estimated factors through t and tracking the post-formation $t + 1$ return.

We report results in Table 2.4. All conditional factor specifications (IPCA and CA_0 through CA_3) produce high unconditional Sharpe ratio statistics, consistent with the find-

Table 2.3: Out-of-Sample Sharpe Ratios of Long-Short Portfolios

	K					
Equal-Weight	1	2	3	4	5	6
FF	-0.66	-0.85	-0.40	-0.30	0.36	-0.21
PCA	0.28	0.09	0.13	-0.08	-0.12	0.15
IPCA	0.20	0.19	1.26	2.16	2.31	2.25
CA_0	0.23	0.32	1.34	1.87	2.10	2.18
CA_1	0.30	0.39	2.12	2.63	2.67	2.60
CA_2	0.30	0.38	2.16	2.64	2.68	2.63
CA_3	0.31	0.38	2.19	2.57	2.57	2.59
	K					
Value-Weight	1	2	3	4	5	6
FF	-0.82	-1.13	-0.69	-0.60	0.18	-0.53
PCA	0.12	-0.18	0.05	-0.10	-0.30	-0.08
IPCA	-0.15	-0.07	0.59	0.81	1.05	0.96
CA_0	-0.11	-0.03	0.41	0.81	0.83	0.88
CA_1	-0.03	0.11	0.91	1.30	1.48	1.40
CA_2	-0.03	0.08	0.92	1.39	1.45	1.53
CA_3	-0.02	0.08	1.09	1.41	1.34	1.51

Note: In this table, we report annualized out-of-sample Sharpe ratios for long-short portfolios using Fama-French models (FF), a vanilla factor model (2.5), and a variety of conditional autoencoders, CA_0, CA_1, CA_2, CA_3, based on (2.9), respectively, where the number of factors in (2.5) or the number of neurons in the hidden layer on the right-hand side of (2.9), K, varies from 1 to 6.

ings of KPS. The most dominant overall model on this dimension is CA_3 with five factors, though performance is broadly similar for CA_1 through CA_3. Static factor models perform markedly worse than conditional models. Results in Table 2.4 reflect the fact that conditional models capture extensive comovement among assets while, at the same time, reconciling their differences in average returns with their factor loadings. These results should not be viewed as performance of implementable trading strategies. The factor tangency portfolios describe the mean-variance efficiency of models without considering practical frictions such as trading costs. Instead, they should be viewed as providing a non-implementable but nonetheless helpful quantitative comparison of models' mean-variance efficiency in economic terms.

2.3.5 Risk Premia vs. Mispricing

An important implication emerges from a comparison of Table 2.2 versus the return prediction analysis of Gu et al. [2019a]. In their paper, the best performing machine learning model forecasts monthly individual stock returns (in the exact same data set as ours) with an R^2 of 0.40%. Yet theirs are pure prediction models—there is no factor structure or risk-return tradeoff—and thus they make no distinction between predictability coming through compen-

Table 2.4: Out-of-Sample Factor Tangency Portfolio Sharpe Ratios

	K					
	1	2	3	4	5	6
FF	0.51	0.41	0.53	0.71	0.71	0.82
PCA	0.35	0.23	0.25	0.38	0.48	0.55
IPCA	0.39	0.44	1.81	3.14	3.71	3.72
CA_0	0.42	0.48	1.47	1.76	1.94	1.97
CA_1	0.56	0.91	3.18	3.82	3.63	4.58
CA_2	0.54	0.75	3.56	4.26	4.72	2.77
CA_3	0.54	0.77	3.94	4.75	4.94	4.37

Note: In this table, we report annualized out-of-sample Sharpe ratios for the mean-variance efficient portfolio of factors using observable factor models (FF), PCA, IPCA, and conditional autoencoders CA_0 through CA_3. In all cases, the number of factors K varies from 1 to 6. We scale the tangency weights each month by targeting 1% monthly volatility based on historical estimates of factor risk premium and covariance matrix.

sation for risk exposure, and compensation from mispricing (i.e., alpha). In contrast, the nonlinear factor models in this paper force all the characteristic-based predictability to come solely through factor risk exposures. That is, the conditional autoencoder models are all specified without an intercept, thus *they impose the economic restriction of no-arbitrage*. Despite this restriction, the conditional autoencoder model achieves nearly identical predictive power for monthly stock returns, 0.58% for the CA_2 specification. This is a significant result—it suggests that stock characteristics predict returns not because they capture "anomalous" compensation without risk, but rather because the characteristics proxy for (and help identify) compensated factor risk exposures.

In this section, we directly test whether the zero-intercept no-arbitrage restriction is satisfied in the data. If it is, the time series average of model residuals for each asset—that is, the pricing errors in our model—should be statistically indistinguishable from zero. We focus this analysis on unconditional pricing errors, defined as:

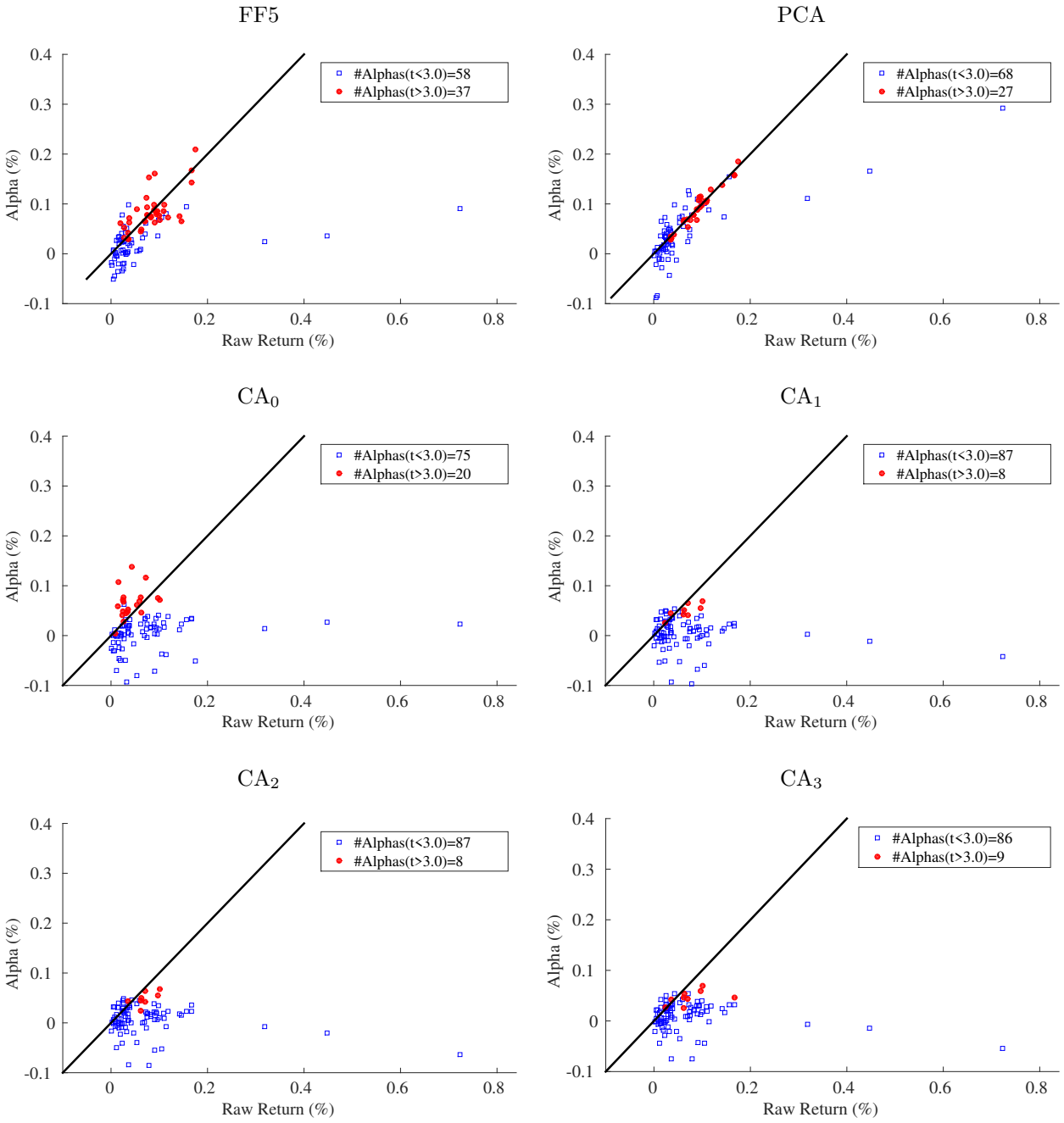
$$\alpha_i := \mathrm{E}(u_{i,t}) = \mathrm{E}(r_{i,t}) - \mathrm{E}(\beta'_{i,t-1} f_t).$$

Alphas for the managed portfolios x_t are defined analogously.

We focus our pricing error tests on x_t, whose comparatively low dimensionality avoids inferential difficulties that arise with r_t.[16] To construct estimates of the out-of-sample pricing error, we calculate the average difference between x_t and its out-of-sample model fit. These pricing errors can be interpreted as the average gain of a hedging portfolio that has a zero-exposure on any systematic factors. In a no-arbitrage model, zero-exposure assets should earn

16. For example, stock-level idiosyncratic risk is so large that stock-level alpha estimates tend to be extremely noisy.

Figure 2.3: Out-of-sample Pricing Errors Across Models

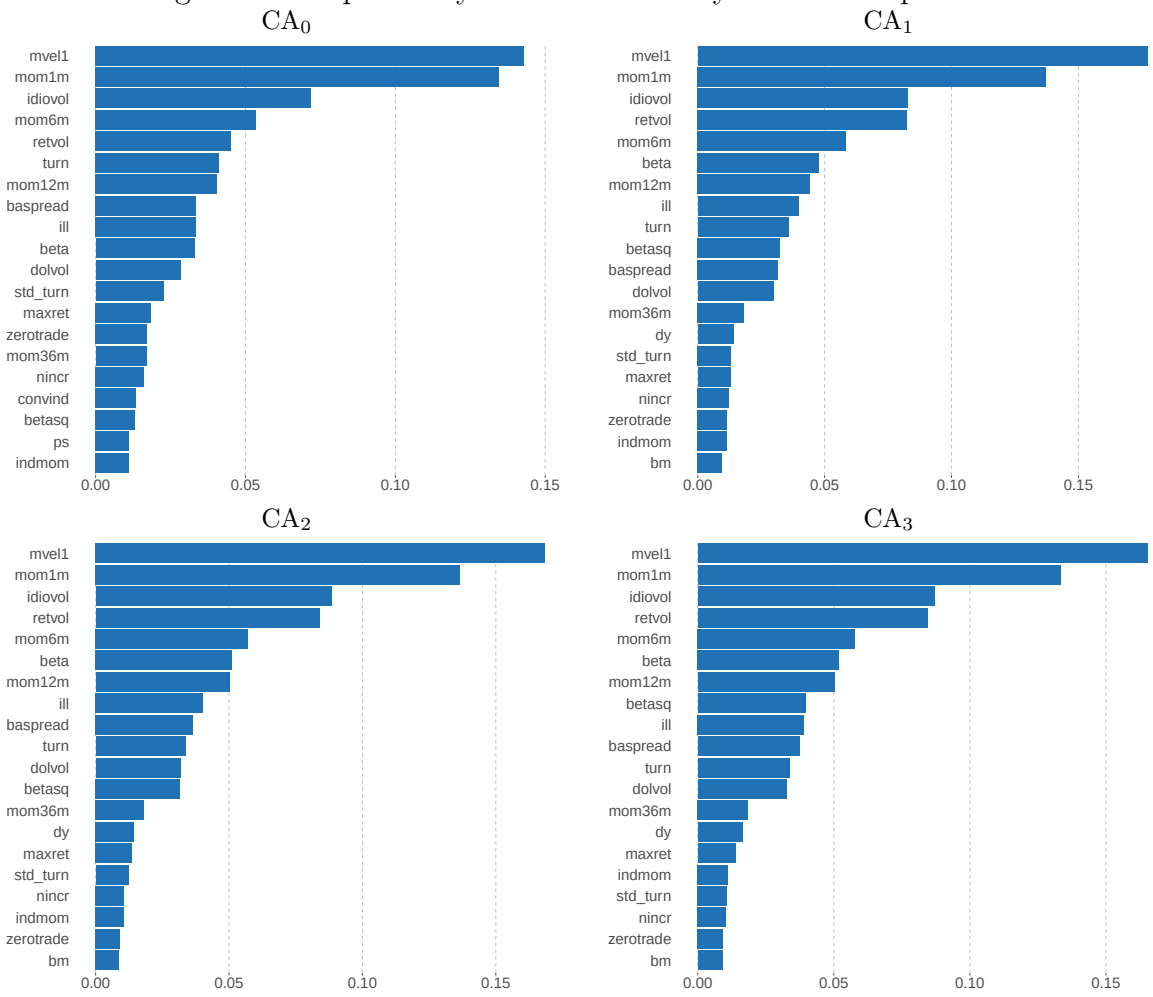


Note: The figure reports out-of-sample pricing errors (alphas) for 95 characteristic-managed portfolios x_t, relative to the Fama-French five-factor model (FF5), the static linear latent five-factor model (PCA), and conditional autoencoders (CA_0 through CA_3). Alphas with t-statistics in excess of 3.0 are shown in red dots, while insignificant alphas are shown in hollow squares.

zero excess return.

Figure 2.3 scatters the estimated out-of-sample pricing errors for each model against the

Figure 2.4: Top Twenty Characteristics by Variable Importance



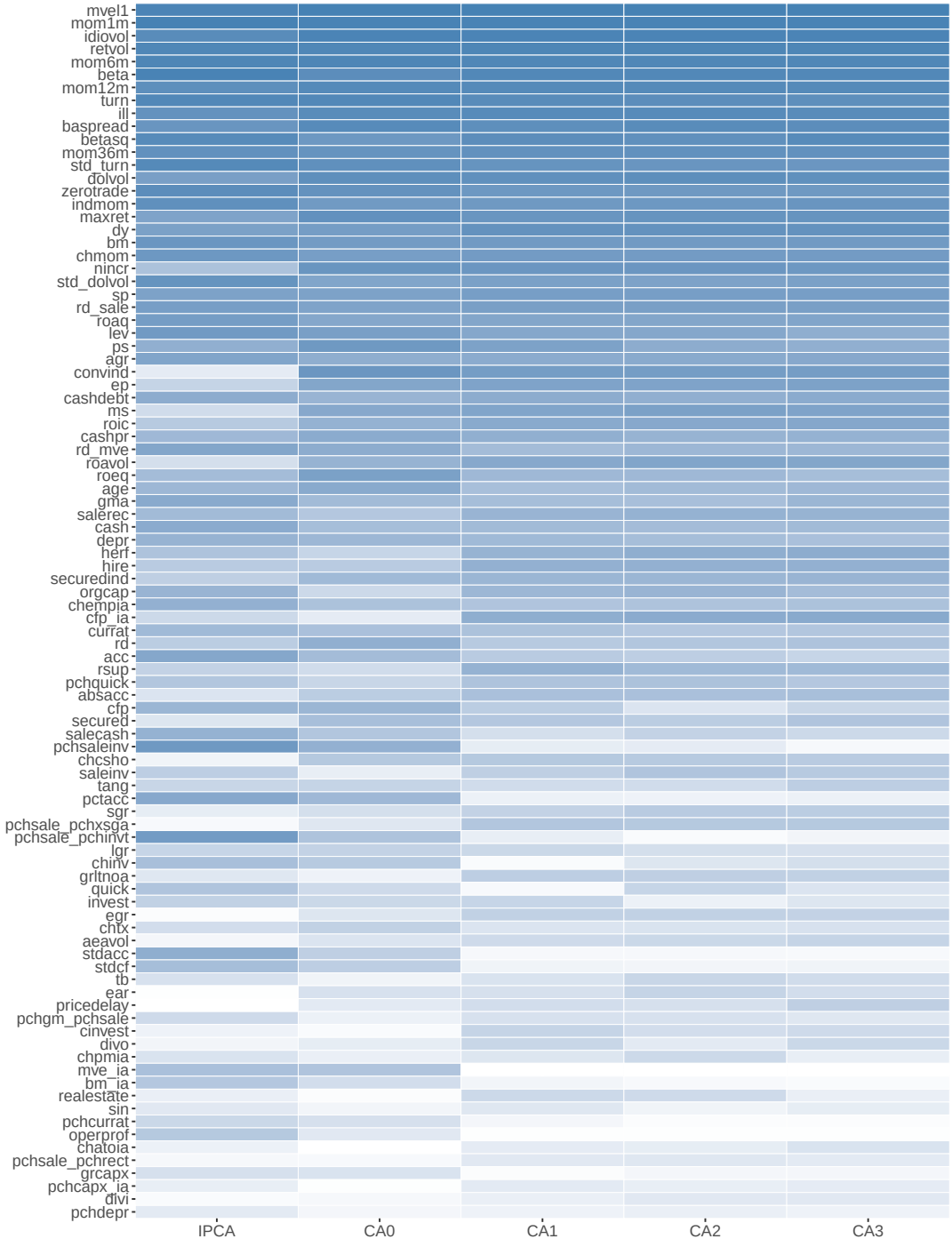
Note: This figure compares variable importance for the top twenty most influential variables in each model, based on an average over all training samples. The variable importances within each model are normalized to sum to one. All models fix $K = 5$.

average returns of x_t. The figure also reports the number of alphas whose t-statistics exceed 3.0. The overall magnitude of alphas shrinks as we move from static linear models to nonlinear conditional autoencoders. For the five-factor Fama-French model, 37 of the 95 managed portfolios have alpha t-statistics in excess of 3.0. For CA_2, that number drops to 8 out of 95. Furthermore, those that remain significant are economically small (below 7 basis points per month) compared to alphas from the Fama-French model.

2.3.6 Characteristics Importance

Following Gu et al. [2019a], we identify influential covariates by ranking them according to a notion of variable importance, defined as the reduction in total R^2 resulting from setting

Figure 2.5: Overall Importance Rankings of All Characteristics



Note: This figure ranks 94 stock-level characteristics 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 top and least influential on bottom. Columns correspond to individual models, and color gradients within each column indicate the most influential (dark blue) to least influential (white) variables.

Figure 2.6: Separate Importance Rankings of All Characteristics

Note: These two plots rank 94 stock-level characteristics in terms of model contribution to $\beta(z_{i,t-1})$ and f_t, respectively. Characteristics are ordered according to the same order in Figure 2.5. Columns correspond to individual models, and color gradients within each column indicate the most influential (dark blue) to least influential (white) variables.

all values of a given characteristic to zero while holding the remaining model estimates fixed. For this analysis, we focus on the five-factor specification of each model.

Figures 2.4 illustrates characteristic importance for each conditional autoencoder specification. It focuses on the top 20 characteristics for each model. Beyond these, variable importance hovers near zero (we show importance for the full list of characteristics in Figure 2.5). The total contribution by the top twenty characteristics is around 80% for CA_0, and 90% for CA_1 through CA_3.

Three categories of characteristics stand out as the most influential. The first is a price trend category, which includes short-term reversal (mom1m), stock momentum (mom12m), momentum change (chmom), industry momentum (indmom), recent maximum return (maxret), and long-term reversal (mom36m). The second category includes liquidity variables, such as turnover and turnover volatility (turn, std_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). Interestingly, all variants of the autoencoder model agree on the importance of these three categories. Moreover, these results closely coincide with the findings of Gu et al. [2019a], who track variable importance using R^2_{pred} (there is no notion of R^2_{total} in their analysis because they focus solely on prediction and therefore do not consider contemporaneous factor associations). This consistency of variable importance across different objectives is an indication of robustness in our list of key variables, and an indication that these variables matter for understanding variation in both expected returns and realized returns.

We further look into the importance of characteristics for the beta and factor networks separately, in Figure 2.6. To calculate the characteristics importance for the beta (resp. factor) network, we again set all values of a given characteristic in the beta (resp. factor) network to zero, without altering the values of this characteristic in the factor (resp. beta) network, and then measure the reduction in total R^2. Interestingly, the relative importance of characteristics are consistent for the two networks.

2.3.7 Robustness Check

Last but not least, we demonstrate the robustness with respect to the choice of assets in the training and testing samples. In particular, we re-train the CA_2 model using subsamples of stocks comprised of odd or even permnos, respectively. We report the out-of-sample total $R^2(\%)$, predictive $R^2(\%)$, equal-weight and value-weight Sharpe ratios for the subsamples in Table 2.5. Throughout, the CA_2 model performs almost equally well, even when the assets used in the training and testing samples are completely non-overlapped.

Table 2.5: Using Subsamples of Stocks Split by Odd or Even Permnos

	Training Sample			
	Total R^2(%)		Predictive R^2(%)	
Testing Sample	Odd	Even	Odd	Even
Odd	13.7	13.6	0.48	0.49
Even	13.6	13.5	0.52	0.54

	Equal-Weight SR		Value-Weight SR	
Testing Sample	Odd	Even	Odd	Even
Odd	2.42	2.38	1.28	1.26
Even	2.52	2.53	1.29	1.19

Note: In this table, we report the out-of-sample total R^2(%), predictive R^2(%), equal-weight and value-weight Sharpe ratios for subsamples of stocks that have odd and even permnos, based on parameters estimated separately with each subsample, respectively. In total there are 29,892 unique tickers in our dataset including 14,984 tickers with odd permnos and 14,908 tickers with even permno. Columns indicate the subsample (Odd, Even) for which we estimate parameters, whereas rows represent the subsample (Odd, Even) for which we evaluate the OOS performance. All estimates are based on the five-factor CA_2 model.

2.4 Monte Carlo Simulations

To demonstrate the finite sample performance of our autoencoder learning method, we simulate a conditional 3–factor model for excess returns r_t, for $t = 1, 2, \ldots, T$:

$$r_{i,t} = \beta_{i,t-1} f_t + \varepsilon_{i,t}, \quad \beta_{i,t-1} = g^*(c_{i,t-1}; \theta), \quad f_t = W x_t + \eta_t$$

where c_t is an $N \times P_c$ matrix of characteristics, f_t is a 3×1 vector of factors, x_t is a $P_x \times 1$ vector of factor components, W is a $3 \times P_x$ factor weighting matrix, and η_t and ε_t are 3×1 and $N \times 1$ vectors of idiosyncratic errors respectively.

We choose $x_t \sim \mathcal{N}(0.03, 0.1^2 \times \mathbb{I}_{P_x})$, $\eta_t \sim \mathcal{N}(0, 0.01^2 \times \mathbb{I}_3)$ and $\varepsilon_{i,t} \sim t_5(0, 0.1^2)$, in which their variances are calibrated so that the average time series R^2 is about 45% and the average annualized volatility is around 60%.

We simulate the panel of characteristics for each $1 \leq i \leq N$ and each $1 \leq j \leq P_c$ from the following model:

$$c_{ij,t} = \frac{2}{n+1} \text{rank}(\bar{c}_{ij,t}) - 1, \quad \bar{c}_{ij,t} = \rho_j \bar{c}_{ij,t-1} + \epsilon_{ij,t}, \tag{2.22}$$

where $\rho_j \sim \mathcal{U}[0.9, 1]$, and $\epsilon_{ij,t} \sim \mathcal{N}(0, 1)$, so that the characteristics feature some degree of persistence over time, yet is cross-sectionally normalized to be within $[-1, 1]$. This matches our data cleaning procedure in the empirical study.

We consider one fixed matrix W and two cases of $g^\star(\cdot)$ functions:

$$W = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}$$

(a) Linear factor loadings: $g^*(c_{i,t};\theta) = (1.2 \times c_{i1,t}, c_{i2,t}, 0.8 \times c_{i3,t})'$.

(b) Non-linear factor loadings: $g^*(c_{i,t};\theta) = \left(c_{i1,t}^2, 2 \times (c_{i1,t} \times c_{i2,t}), 0.6 \times \text{sgn}(c_{i3,t})\right)'$.

Throughout, we fix $N = 200$, $T = 180$ and $P_c = P_x = 50$. In both cases, $g^\star(\cdot)$ only depends on 3 covariates, so there are only 3 non-zero entries in θ, denoted as θ_0. Case (a) is simple and sparse linear model. Case (b) involves a nonlinear covariate $c_{i1,t}^2$, a nonlinear and interaction term $(c_{i1,t} \times c_{i2,t})$, and a dummy variable $\text{sgn}(c_{i3,t})$. We calibrate the values of θ_0 such that the total R^2 is around 40%, and the predictive R^2 is 5%.

For each Monte Carlo sample, we divide the whole time series into 3 consecutive subsamples of equal length for training, validation, and testing, respectively. For Vanilla PCA and IPCA, we combine training and validation samples together because they don't have any tuning parameter. For CA_0, CA_1, CA_2 and CA_3 we estimate them in the training sample, then choose tuning parameters for each method in the validation sample, and calculate the prediction errors in the testing sample.

We report the average OOS total and predictive R^2s for each method over 100 Monte Carlo repetitions in Table 2.6. For model (a), IPCA delivers the best OOS total and predictive R^2s. This is not surprising given that the true model is sparse and linear in the input covariates. More advanced methods such as CA_1, CA_2 and CA_3 tend to overfit, so their performance is slightly worse. By contrast, for model (b), these methods clearly beat IPCA, because the latter cannot capture the nonlinearity in the model. The Vanilla PCA method is always overfitting in the training sample so that it cannot achieve a good OOS R^2s. The comparison among autoencoder models demonstrates a stark trade-off between model flexibility and implementation difficulty. As shown in the table, shallower conditional autoencoders tend to outperform in our simulation setting, which is consistent with our findings in empirical analysis.

Overall, the simulation results suggest that the conditional autoencoder methods are successful in learning the factor structure in both linear and nonlinear situations. This is not surprising, as these methods are implemented to improve fitting and prediction by allowing more complex functional forms of conditional factor loadings.

2.5 Conclusion

We propose a new approach to latent factor modeling for asset pricing that draws on autoencoder methods from the machine learning literature. We adapt the standard autoencoder

Table 2.6: Comparison of Total R^2(%)s and Predictive R^2(%)s in Simulations

Model (a)	K					
Total. R^2	1	2	3	4	5	6
PCA	3.5	4.7	5.5	6.3	7.1	7.8
IPCA	18.6	32.2	40.7	41.0	41.4	41.7
CA_0	15.6	26.7	33.7	33.5	33.4	33.2
CA_1	17.6	30.3	38.1	37.7	37.3	37.1
CA_2	17.7	29.2	36.8	36.5	36.3	35.9
CA_3	17.6	25.6	30.0	29.5	26.3	23.4
Pred. R^2						
PCA	0.17	0.10	0.04	0.01	-0.01	-0.03
IPCA	2.20	2.93	3.33	3.32	3.32	3.32
CA_0	2.04	2.84	3.17	3.14	3.12	3.13
CA_1	2.11	2.93	3.27	3.29	3.26	3.26
CA_2	2.10	2.85	3.22	3.22	3.23	3.22
CA_3	2.06	2.57	2.89	2.86	2.58	2.39

Model (b)	K					
Total. R^2	1	2	3	4	5	6
PCA	3.4	5.1	6.0	6.6	7.3	7.9
IPCA	11.0	11.4	11.9	12.3	12.7	13.1
CA_0	8.5	8.2	7.9	7.6	7.4	7.2
CA_1	15.0	24.6	31.8	32.0	31.9	31.8
CA_2	15.7	23.5	30.9	31.8	30.2	28.2
CA_3	15.9	15.6	14.6	14.0	11.2	9.2
Pred. R^2						
PCA	0.15	0.19	0.15	0.12	0.10	0.09
IPCA	0.84	0.82	0.81	0.80	0.79	0.79
CA_0	0.80	0.76	0.77	0.76	0.72	0.70
CA_1	1.83	2.31	2.70	2.70	2.71	2.73
CA_2	1.95	2.24	2.73	2.80	2.69	2.53
CA_3	1.77	1.43	1.32	1.26	1.06	0.86

Note: In this table, we report the average out-of-sample (OOS) Total R^2(%)s and Predictive R^2(%)s for models (a) and (b) using PCA, IPCA, CA_0, CA_1, CA_2 and CA_3, respectively. We fix $N = 200$, $T = 180$, and $P_c = P_x = 50$. The number of Monte Carlo repetitions is 100.

to allow latent factors and factor exposures to depend on asset characteristic conditioning variables. The result is a nonlinear conditional asset pricing model that embeds the economic restriction of no-arbitrage within a broader neural network framework.

In the empirical context of monthly US stock returns, our conditional autoencoder model dominates competing asset pricing models, including Fama-French models, PCA methods, and linear conditioning methods such as IPCA. A long-short decile spread portfolios sorted on stock return predictions from our preferred autoencoder produces an annualized value-weighted Sharpe ratio of 1.53, beating the next closest competitor (IPCA, with Sharpe ratio

0.96) by a wide margin, and on a purely out-of-sample basis. Finally, the pricing errors in our model (likewise measured on an out-of-sample basis) are a fraction of the magnitude of those from traditional Fama-French factor models.

2.6 Appendix

2.6.1 Mathematical Proofs

Notation

We use $(A : B)$ to denote the column concatenation of two matrices A and B. The vector e_i has a value of 1 in the i^{th} position and 0 elsewhere (and whose dimension implicitly conforms to the context). Likewise, the vector ι denotes a conformable vector with all entries being 1. For any time series of vectors $\{a_t\}_{t=1}^T$, we denote $\bar{a} = \frac{1}{T}\sum_{t=1}^T a_t$. In addition, we write $\bar{a}_t = a_t - \bar{a}$. A denotes the matrix $(a_1 : a_2 : \ldots : a_T)$, and $\bar{A} = A - \bar{a}\iota'$ correspondingly. We use $\lambda_j(A)$ to denote the jth largest eigenvalue of A, and $\sigma_j(A)$ the jth largest singular value. We use $\|A\|$ and $\|A\|_{\mathrm{F}}$ to denote the operator norm (or $\mathbb{L}_2$ norm) and the Frobenius norm of a matrix $A = (a_{ij})$, that is, $\sqrt{\lambda_1(A'A)}$ and $\sqrt{\mathrm{Tr}(A'A)}$, respectively.

Proofs

Proof of Proposition 1. At first, we set the partial derivative with respect to $b^{(1)}$ to zero.

$$\begin{aligned}
\frac{\partial}{\partial b^{(1)}} \left\| R - \left(b^{(1)}\iota' + W^{(1)}(b^{(0)}\iota' + W^{(0)}R) \right) \right\|_{\mathrm{F}}^2 &= 0 \\
\left(R - b^{(1)}\iota' - W^{(1)}(b^{(0)}\iota' + W^{(0)}R) \right)\iota &= 0 \\
\widehat{b}^{(1)} = \frac{1}{T}\left(R\iota - TW^{(1)}b^{(0)} - W^{(1)}W^{(0)}R\iota \right).
\end{aligned}$$

Then we insert the solution into (2.8).

$$\begin{aligned}
&\min_{b,W} \left\| R - \left(b^{(1)}\iota' + W^{(1)}(b^{(0)}\iota' + W^{(0)}R) \right) \right\|_{\mathrm{F}}^2 \\
=&\min_{W} \left\| \left(R - \frac{1}{T}R\iota\iota' \right) - W^{(1)}W^{(0)} \left(R - \frac{1}{T}R\iota\iota' \right) \right\|_{\mathrm{F}}^2 \\
=&\min_{W} \left\| \bar{R} - W^{(1)}W^{(0)}\bar{R} \right\|_{\mathrm{F}}^2,
\end{aligned}$$

where $\bar{R}$ is a matrix of demeaned returns. Thus, the problem becomes independent of the bias terms. We focus on the weights $W^{(0)}$and $W^{(1)}$.
Next, we set the partial derivative with respect to $W^{(0)}$ to zero and assume $\bar{R}\bar{R}'$ and $W^{(1)\prime}W^{(1)}$ both have full rank.

$$\frac{\partial}{\partial W^{(0)}} \left\| \bar{R} - W^{(1)}W^{(0)}\bar{R} \right\|_{\mathrm{F}}^2 = 0$$

$$W'^{(1)}W^{(1)}W'^{(0)}\bar{R}\bar{R}' - W'^{(1)}\bar{R}\bar{R}' = 0$$
$$W^{(0)} = (W'^{(1)}W^{(1)})^{-1}W'^{(1)}.$$

So, the optimization problem becomes

$$\begin{aligned}
&\min_{W} \left\| \bar{R} - W^{(1)}W^{(0)}\bar{R} \right\|_{\mathrm{F}}^2 \\
&= \min_{W^{(1)}} \left\| \bar{R} - W^{(1)} \left(W'^{(1)}W^{(1)} \right)^{-1} W'^{(1)}\bar{R} \right\|_{\mathrm{F}}^2 \\
&= \min_{W^{(1)}} \left\| \bar{R} - \mathbb{P}_{W^{(1)}}\bar{R} \right\|_{\mathrm{F}}^2 .
\end{aligned}$$

The Eckart-Young-Mirsky theorem for the Frobenius norm states that the best rank K approximation for a matrix is $\sum_{i=1}^{K} s_i \widehat{p}_i \widehat{q}_i' = \widehat{P}\Lambda\widehat{Q}$. Therefore, $W^{(1)}$ is the solution if and only if it satisfies

$$\mathbb{P}_{W^{(1)}}\bar{R} = \widehat{P}\Lambda\widehat{Q} \tag{2.23}$$

It is obvious that $W^{(1)} = \widehat{P}$ is one solution for the above equation, because

$$\mathbb{P}_{W^{(1)}}\bar{R} = \widehat{P}(\widehat{P}'\widehat{P})^{-1}\widehat{P}'\bar{R} = \widehat{P}(\widehat{P}'\widehat{P})^{-1}\widehat{P}'(\widehat{P}\Lambda\widehat{Q} + \widehat{U}) = \widehat{P}\Lambda\widehat{Q}.$$

However, the solution of (2.23) is not unique. $W^{(1)} = \widehat{P}A$ is also a solution, where A is any $K \times K$ full-rank matrix. The linear autoencoder is equivalent to PCA since they have the same factor loading matrix $\widehat{P}$ (up to a rotation matrix). □

Proof of Proposition 2. At first, we consider the objective function of IPCA when $Z_{t-1}'Z_{t-1} = \Sigma$. The first-order condition for F is given by

$$\widehat{f_t} \;=\; (\Gamma Z_{t-1}'Z_{t-1}\Gamma')^{-1}\Gamma Z_{t-1}'r_t = (\Gamma\Sigma\Gamma')^{-1}\Gamma Z_{t-1}'r_t.$$

Then we plug in $\widehat{f_t}$ to the objective function of IPCA (2.17)

$$\min_{\Gamma,F} \sum_{t=1}^{T} \left\| r_t - Z_{t-1}\Gamma' f_t \right\|^2 = \min_{\Gamma} \sum_{t=1}^{T} \left\| r_t - Z_{t-1}\Gamma'(\Gamma\Sigma\Gamma')^{-1}\Gamma Z_{t-1}'r_t \right\|^2 . \tag{2.24}$$

For the two-sided Autoencoder model, we use the managed portfolios $x_t = (Z_{t-1}'Z_{t-1})^{-1}Z_{t-1}'r_t$

as the inputs of the right-hand side. We can rewrite the objective function (2.18) as

$$\min_{W_0,W_1} \sum_{t=1}^{T} \left\| r_t - Z_{t-1} W_0' W_1 x_t \right\|^2 = \arg\min_{W_0,W_1} \sum_{t=1}^{T} \left\| r_t - (x_t' \otimes Z_{t-1} W_0') \mathrm{vec}(W_1) \right\|^2 . \quad (2.25)$$

Next the first order condition of $\mathrm{vec}(W)$ is

$$\begin{aligned} \mathrm{vec}(W_1) &= \left(\sum_{t=1}^{T} (x_t' \otimes Z_{t-1} W_0')' (x_t' \otimes Z_{t-1} W_0') \right)^{-1} \left(\sum_{t=1}^{T} (x_t' \otimes Z_{t-1} W_0')' r_t \right) \\ &= \left(\sum_{t=1}^{T} x_t x_t' \otimes W_0 Z_{t-1}' Z_{t-1} W_0' \right)^{-1} \left(\sum_{t=1}^{T} \left(x_t \otimes W_0 Z_{t-1}' r_t \right) \right) . \end{aligned}$$

We plug in $Z_{t-1}' Z_{t-1} = \Sigma$,

$$\begin{aligned} \mathrm{vec}(W_1) &= \left(\sum_{t=1}^{T} x_t x_t' \otimes W_0 \Sigma W_0' \right)^{-1} \left(\sum_{t=1}^{T} \left(x_t \otimes W_0 Z_{t-1}' r_t \right) \right) \\ &= \left(\left(\sum_{t=1}^{T} x_t x_t' \right)^{-1} \otimes (W_0 \Sigma W_0')^{-1} \right) \left(\sum_{t=1}^{T} \left(x_t \otimes W_0 Z_{t-1}' r_t \right) \right) \\ &= \sum_{t=1}^{T} \left(\left(\sum_{t=1}^{T} x_t x_t' \right)^{-1} x_t \otimes (W_0 \Sigma W_0')^{-1} W_0 Z_{t-1}' r_t \right) . \end{aligned}$$

Then we recover W_1 from $\mathrm{vec}(W_1)$

$$\begin{aligned} W_1 &= \sum_{t=1}^{T} \left[(W_0 \Sigma W_0')^{-1} W_0 Z_{t-1}' r_t x_t' \left(\sum_{t=1}^{T} x_t x_t' \right)^{-1} \right] \\ &= \sum_{t=1}^{T} \left[(W_0 \Sigma W_0')^{-1} W_0 \Sigma x_t x_t' \left(\sum_{t=1}^{T} x_t x_t' \right)^{-1} \right] \\ &= (W_0 \Sigma W_0')^{-1} W_0 \Sigma. \end{aligned}$$

We can plug in the solution of W_1 to the objective function of the two-sided Autoencoder

(2.18),

$$\min_{W_0,W_1} \sum_{t=1}^{T} \left\| r_t - Z_{t-1} W_0' W_1 x_t \right\|^2 = \arg\min_{W_0} \sum_{t=1}^{T} \left\| r_t - Z_{t-1} W_0' (W_0 \Sigma W_0')^{-1} W_0 \Sigma x_t \right\|^2. \quad (2.26)$$

We see IPCA and the two-sided Autoencoder have the same objective functions 2.24 and 2.26. The Autoencoder solution and the IPCA solution are identical with $\Gamma = W_0$. They give us identical factor estimates $\hat{f}_t$ and factor loading estimates $\hat{\beta}_{i,t-1}$.

□

2.6.2 Algorithms

Algorithm 8: Early Stopping

Initialize $j = 0$, $\epsilon = \infty$ and select the patience parameter p.
while $j < p$ **do**
  Update θ using the training algorithm (e.g., the steps inside the while loop of Algorithm 9 for h steps).
  Calculate the prediction error from the validation sample, denoted as ϵ'.
  if $\epsilon' < \epsilon$ **then**
    $j \leftarrow 0$.
    $\epsilon \leftarrow \epsilon'$.
    $\theta' \leftarrow \theta$.
    else
    $j \leftarrow j + 1$.
  end
end
Result: The final parameter estimate is θ'.

Algorithm 9: Adam for Stochastic Gradient Descent (SGD)

Initialize the target parameter vector θ_0.
Set $m_0 = 0, v_0 = 0, t = 0, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$.
Tuning parameters: learning rate α, batch size b.
while θ_t *not converged* **do**
  $t \leftarrow t + 1$.
  $g_t \leftarrow \nabla_\theta \left[\frac{1}{b} \sum_{s \in B_t} \mathcal{L}(\theta; s) \right] \big|_{\theta=\theta_{t-1}}$, where B_t is the set of batch subsamples.
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$.
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t \odot g_t$.[17]
  $\hat{m}_t \leftarrow m_t / (1 - (\beta_1)^t)$.
  $\hat{v}_t \leftarrow v_t / (1 - (\beta_2)^t)$.
  $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t \oslash (\sqrt{\hat{v}_t} + \epsilon)$.
end
Result: The final parameter estimate is θ_t.

Algorithm 10: Batch Normalization (for one Activation over one Batch)

Input: Values of x for each activation over a batch $\mathcal{B} = \{x_1, x_2, \ldots, x_N\}$.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta := \mathrm{BN}_{\gamma,\beta}(x_i)$$

Result: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i) : i = 1, 2 \ldots, N\}$.

2.6.3 201701-202012 Out-of-Sample Performance

We do the exactly same empirical analysis using the updated data (including firm characteristic data). For the new 48 months (201701-202012) in OOS period, we still implement recursive strategy when training/validation/testing. The prediction performance (total R^2 and predictive R^2) and Sharp ratios of long-short portfolios are reported in the following tables.

We can see that all the autoencoder models and IPCA achieve high R^2s, however the predictive R^2s are tiny when comparing with the previous 30 years OOS period. Additionally, the Sharp rations of the equally weighted long-short portfolios are much higher than value weighted portfolios. The IPCA model is the most robust one when checking both equally-weighted and value-weighted. The autoencoder models get higher equally-weighted SR but the value-weighted SR is lower. In recent 4 years, the margin of machine learning techniques becomes weak. It seems the best strategy for the investors is just to buy and hold the market.

Table 2.7: 201701-202012 Out-of-Sample $R^2_{\text{total}}(\%)$ Comparison

		K					
Model	Test Assets	1	2	3	4	5	6
IPCA	r_t	13.5	14.5	15.1	15.4	15.7	15.8
	x_t	88.8	94.8	96.5	97.0	97.5	97.7
CA_0	r_t	13.1	14.2	14.8	15.0	15.1	15.2
	x_t	87.1	93.6	95.9	96.2	96.2	96.4
CA_1	r_t	13.5	14.6	15.0	15.4	15.7	15.8
	x_t	88.8	94.3	95.3	96.0	96.6	96.7
CA_2	r_t	13.4	14.6	15.0	15.5	15.7	15.8
	x_t	89.1	94.2	95.4	96.1	96.5	96.8
CA_3	r_t	13.4	14.6	15.0	15.5	15.7	15.7
	x_t	88.9	93.9	95.3	96.4	96.6	96.7

Note: In this table, we report the out-of-sample total $R^2(\%)$ for individual stocks r_t and managed portfolios x_t using IPCA, and conditional autoencoders CA_0 through CA_3. In all cases, the number of factors K varies from 1 to 6.

Table 2.8: 201701-202012 Out-of-Sample $R^2_{\text{pred}}(\%)$ Comparison

		K					
Model	Test Assets	1	2	3	4	5	6
IPCA	r_t	0.25	0.25	0.12	0.12	0.15	0.14
	x_t	1.52	1.55	1.30	0.97	1.10	1.05
CA_0	r_t	0.26	0.26	0.19	0.21	0.19	0.20
	x_t	1.54	1.54	1.44	1.23	1.15	1.22
CA_1	r_t	0.24	0.25	0.11	0.15	0.10	0.08
	x_t	1.46	1.59	1.12	0.88	0.76	0.70
CA_2	r_t	0.25	0.27	0.12	0.14	0.19	0.15
	x_t	1.51	1.61	1.25	0.82	0.99	0.92
CA_3	r_t	0.25	0.25	0.14	0.15	0.12	0.10
	x_t	1.54	1.48	1.33	0.87	0.58	0.65

Note: In this table, we report the out-of-sample predictive $R^2(\%)$ for individual stocks r_t and managed portfolios x_t using IPCA, and conditional autoencoders CA_0 through CA_3. In all cases, the number of factors K varies from 1 to 6.

Table 2.9: 201701-202012 Out-of-Sample Sharpe Ratios of Long-Short Portfolios

	K					
Equal-Weight	1	2	3	4	5	6
IPCA	0.36	0.37	-0.42	0.43	1.27	1.27
CA_0	0.41	0.42	-0.43	0.89	1.09	1.16
CA_1	0.30	0.22	0.00	1.46	1.50	1.15
CA_2	0.35	0.47	0.33	1.19	1.84	1.73
CA_3	0.39	0.54	-0.04	0.96	1.36	1.84
	K					
Value-Weight	1	2	3	4	5	6
IPCA	0.43	0.61	-0.85	-0.31	0.83	0.80
CA_0	0.50	0.97	-0.78	0.09	0.73	0.68
CA_1	0.51	0.64	-0.52	0.52	0.61	0.09
CA_2	0.69	0.76	-0.29	-0.53	0.46	0.64
CA_3	0.37	0.75	-0.52	0.35	0.45	0.64

Note: In this table, we report annualized out-of-sample Sharpe ratios for long-short portfolios using IPCA and a variety of conditional autoencoders, CA_0, CA_1, CA_2, CA_3, based on (2.9), respectively, where the number of factors K, varies from 1 to 6.

CHAPTER 3
RETURN PREDICTION WITH TEXT DATA

Research in collaboration with Bryan Kelly and Dacheng Xiu.

We perform a novel analysis of Natural Language Processing methods for the problem of empirical asset pricing: predicting stock returns with the information of News stories. We find the News stories have significant prediction power for equity returns in most of markets throughout the world. We also do a comprehensive comparison between NLP models on different markets with different languages. We demonstrate the advanced NLP methods (word2vec, BERT) have a better understanding on the unstructured data (News stories) and achieve better performance on the return prediction task, which leads to large economic gain on the performance of long-short portfolios.

Key words: Natural Language Processing, Big Data, Return Prediction, Cross-Section of Returns, Daily Long-short Portfolios, Fintech

3.1 Methodology

This section describes two NLP methods: word2vec and BERT that we use in our analysis. The modern NLP methods are only used to extract useful features from text data (alslo known as embeddings). We treat these embeddings as independent variables and do next step modeling based on them. The NLP methods help to extract information from high-dimension unstructured data and statistical models (e.g. regressions, classifications) help to use these embeddings to model the future stock returns.

3.1.1 Word2vec

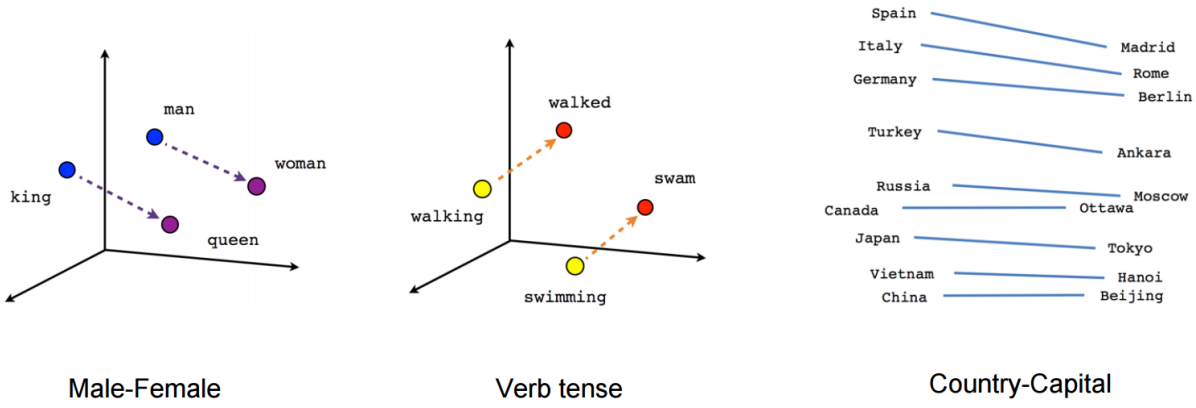
In Natural Language Processing, we often map words or sentences into numerical vectors so that machines can understand it and do complex analysis. Word embedding is a type of mapping that allows words with similar meaning to have similar numerical representation. Word2vec is a popular word-mapping method used in many NLP applications (see Mikolov et al. [2013]).

A traditional way of representing words is using one-hot vector. The length of the vector is equal to the size of the total unique vocabulary in the corpora. So the one-hot embedding of an article is a long vector with 0s and 1s, where 1 element means the article contains the corresponding word and 0 means it doesn't include the word. Although this one-hot representation of words is simple and easy to implement, there are several issues. First, you cannot infer any relationship between two words given their one-hot representation. For instance, the word "increase" and "soar", although they have similar meaning in most of news stories, their targets "1" are far from each other. In addition, sparsity is another issue as there are numerous redundant "0" in the vectors.

Word2vec is an efficient solution to these problems, which leverages the context of the target words. Essentially, we want to use the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation (embeddings). There are a bunch of available pre-trained word2vec models, and in this paper we implement FastText which is an open-source, free, lightweight library supported by Facebook. Most importantly, FastText supports a multiple of languages so that we could use it on News stories from different countries.

Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words). For instance, the tri-grams for the word apple is app, ppl, and ple (ignoring the starting and ending of boundaries of words). The word embedding vector for apple will be the sum of all these n-grams. After training the Neural Network, we will have word embeddings for all the n-grams given the training dataset. Rare words can now be

Figure 3.1: Word2vec example



properly represented since it is likely that some of their n-grams also appears in other words. See Bojanowski et al. [2016],Joulin et al. [2016b] and Joulin et al. [2016a]. The figure 3.1 visualizes the word2vec embeddings of some example words.

We use the FastText package in our empirical study. For each language, the package contains a mapping (function f_w) which maps each of the words in the given language vocabulary to a 300-length vector. So we can define the article embeddings by averaging all words in that article excluding some stop words and words not in the given vocabulary.

$$\text{Word2vec Embedding}(article) = \frac{1}{N} \sum_{word \in article} f_w(word),$$

where $article$ is a list of words (without replacement) that appear in the certain article, with valid mapping vectors and are not stop words. N is the length of the list $article$. The final output Word2vec Embedding($article$) is a numerical vector (300-dimension).

3.1.2 BERT

Although word2vec is a big step forward from one-hot embedding, there are still some drawbacks. First, word2vec only learns the appearance relationship between words but ignores the true context in the article. Second, word2vec cannot understand negation or ambiguity in the News stories.

BERT (Bidirectional Encoder Representations from Transformers) proposed by researchers at Google AI Language has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering, Natural Language Inference, and others. BERT's key technical innovation is applying the bidirectional

training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts (most LSTMs) which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results (Devlin et al. [2018]) show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.

The typical implementation of BERT is Transfer Learning + Fine-tuning to downstream tasks. Now that the OpenAI transformer is pre-trained and its layers have been tuned to reasonably handle language. We won't be bothered by the heavy training computation and only focus on the pre-trained model and applying BERT embedding to our own task of predicting stock returns. In our empirical study, we treat BERT as a feature extraction method. The fine-tuning approach isn't the only way to use BERT. We can use the pre-trained BERT to create contextualized word embeddings, then feed these embeddings to the existing statistical model – a process the paper (Devlin et al. [2018]) shows yield results not far behind fine-tuning BERT on a task such as named-entity recognition. Based on the results of the BERT paper (see figure 3.2), we use the last hidden layer as the contextualized embedding for each word.

Since the length of some News stories is very long (thousands of words), which is not suitable to BERT. Firstly, we split each article into sentences and then use BERT to "transfer" each sentence into numerical embeddings. The final article embeddings are the average of all sentence embeddings in that article.

$$\text{BERT Embedding}(article) = \frac{1}{N_s} \sum_{sentence \in article} \left(\frac{1}{N_w} \sum_{word \in sentence} f_{bert}(word) \right),$$

where N_s is the number of the sentences in *article* and N_w is the number of words in the *sentence*. The final output BERT Embedding(*article*) is a numerical vector (768-dimension).

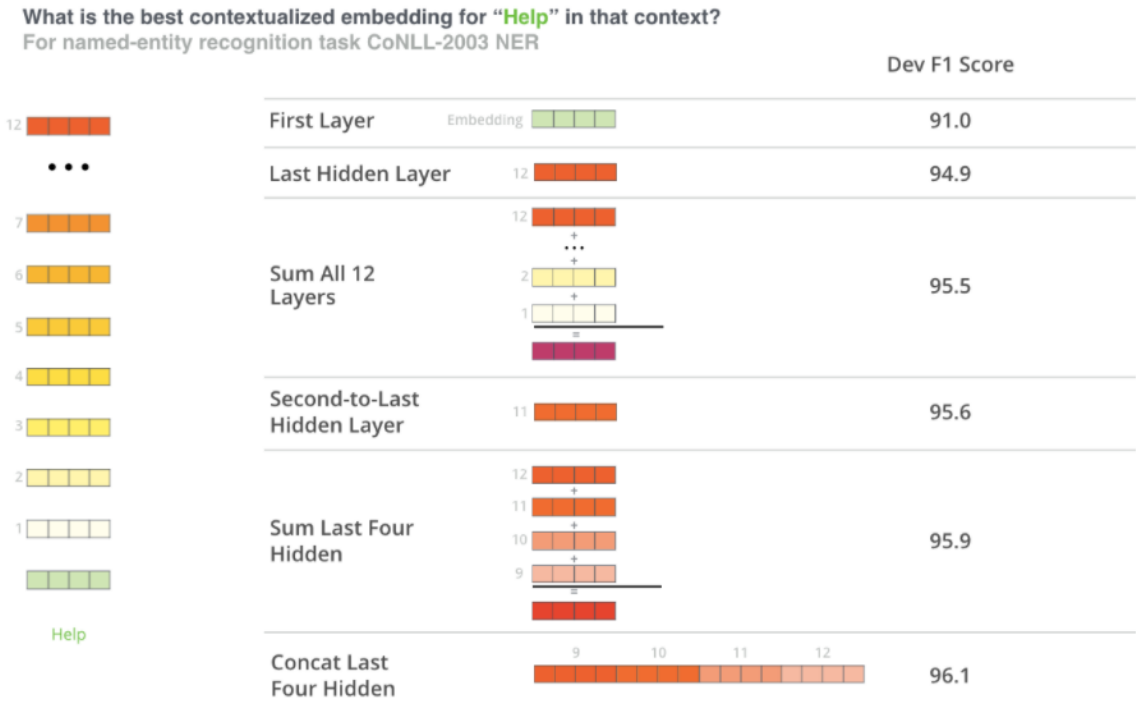
3.1.3 Regressions with Article Embeddings

In our empirical study, each article (corresponded to one unique stock) is a single sample in our model training and testing. For the given article, we can extract word2vec features (X_{word}) and BERT features (X_{bert}) seperately. The response variable for this article is the three-day-compound return $r_{t-1,t+1}$ (following the setting of Ke et al. [2019]).

$$r_{t-1,t+1} = (r_{t-1} + 1)(r_t + 1)(r_{t+1} + 1) - 1,$$

We do three different regressions OLS, Logistic regression and weighted logistics regression (use $|r_{t-1,t+1}|$ as sample weight) with two NLP embeddings. For the benchmark model, we

Figure 3.2: BERT embeddings comparison



refer to Bag-of-Words methodology in the paper Ke et al. [2019] for the comparison.

$$r_{t-1,t+1} \sim \beta_{word} X_{word}$$

$$r_{t-1,t+1} \sim \beta_{bert} X_{bert}$$

3.2 Empirical Study

3.2.1 Data Summary

We used *Thomson Reuters Real-time News Feed and Archive* database in our empirical analysis. The database aggregates many news sources from a majority of countries and regions all over the world. The news stories in the database are in a variety of languages and focus on firms in global financial markets. We follow the filtering methodology in Ke et al. [2019] and analyze articles with only one firm tag. We group all the articles by equity market countries and their corresponding official languages (e.g. US represents the tagged firm is on the list of US equity market and in English). Table 3.1 reports the summary statistics of articles in 16

equity markets and in 13 languages. The real-time news feed is from January 1, 1996 to July 31, 2019, amounting to 15,639,664 unique articles. Other than languages and countries, we extract the local date, local timestamp, tagged firm ticker, headline, and body text of each article. The local date and timestamp are in time zone of the local stock exchange (e.g. US articles map to the Eastern Standard time zone).

Figure 3.3 and 3.5 (in appendix) plot the total number of articles by year and the average number of articles in each half-hour interval throughout the day for each country separately. There is a dramatically increasing of articles starting from 2003. Most of news feeds arrive during day time especially around the market open and close.

Table 3.1: Summary Statistics

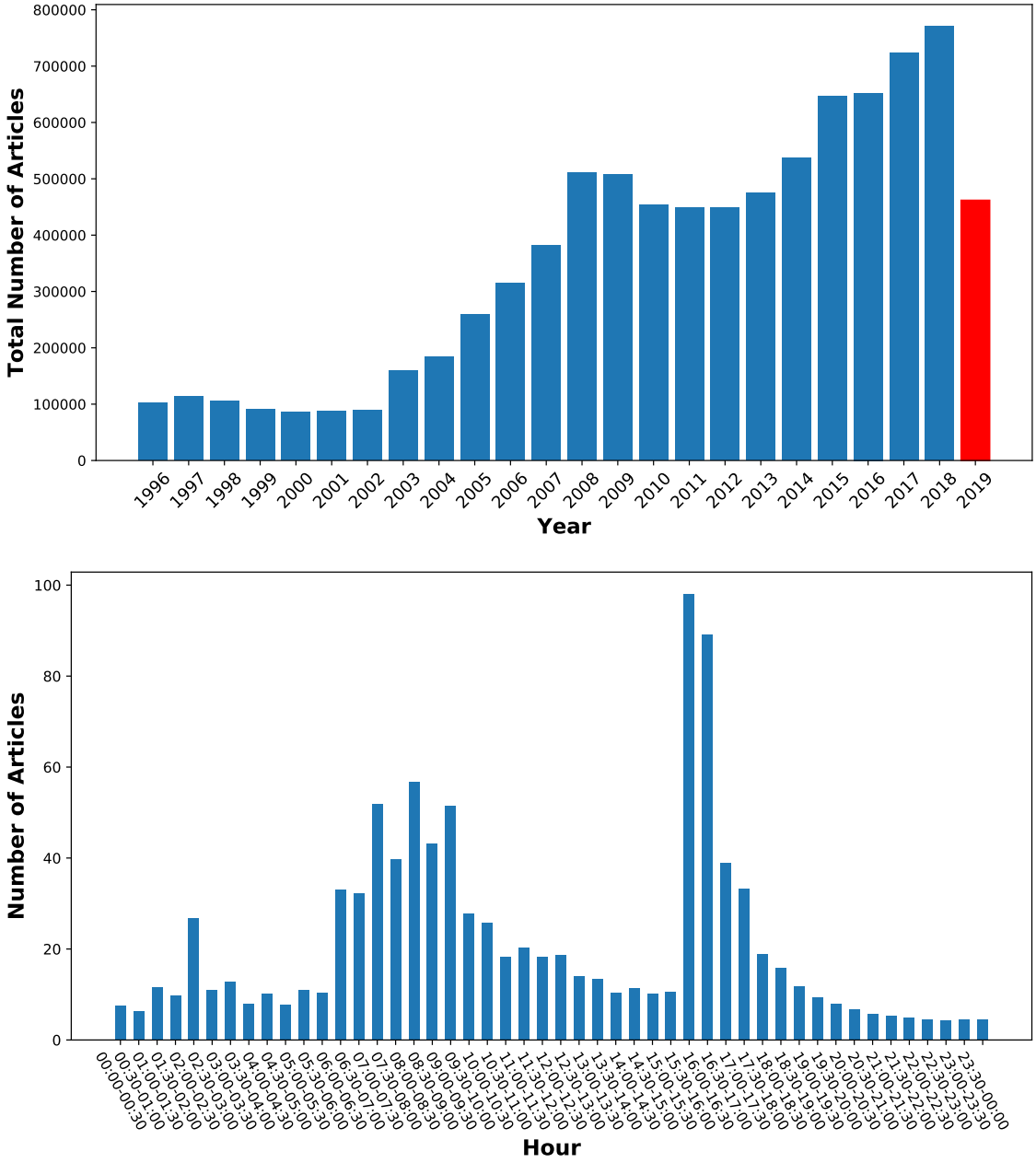
Country	Language	Total News	Start Day	End Day	Total Days	Average News
US	en	8612849	1996-01-01	2019-07-31	8578	1004.1
China	zh	2191824	1996-01-02	2019-08-01	6399	342.5
UK	en	1245548	1996-01-01	2019-07-31	7323	170.1
Australia	en	1055752	1996-01-02	2019-08-01	6590	160.2
Canada	en	843886	1996-01-02	2019-07-31	7028	120.1
Japan	ja	456308	1996-01-04	2019-08-01	6049	75.4
Germany	de	371509	1996-01-02	2019-07-31	7814	47.5
Italy	it	243477	1996-01-04	2019-07-31	6461	37.7
France	fr	241328	1996-01-02	2019-07-31	6960	34.7
Sweden	sv	172156	2001-06-06	2019-07-31	4677	36.8
Denmark	da	71672	1996-01-19	2019-07-31	4611	15.5
Spain	es	61415	1996-01-04	2019-07-31	5791	10.6
Finland	fi	38181	2002-04-24	2019-07-31	4102	9.3
Greece	el	13295	2002-01-29	2019-07-31	3179	4.2
Portugal	pt	12684	1998-06-16	2019-07-31	2960	4.3
Netherlands	nl	7780	1996-01-04	2019-07-30	3176	2.4

Note: In this table, we report the total number of news after our filtering and the sample period range for each country. We also report the total number of days in which there is news coming, and the daily average number of news.

3.2.2 Text Pre-processing

We applied to two different text pre-processing methods for different models. For bag of words model and word2vec model, we need transform the article text to a list of words for

Figure 3.3: Average Article Counts in US



Note: This top figure plots the annual time series of the total number of articles in US from January 1996 to July 2019. We only provide the total number of articles in 7 months for 2019 (highlighted in red). The bottom figure plots the average numbers of articles per hour (local trading time).

further calculation. The pre-processing contains 1. clean special characters(website addresses, email addresses and etc.) 2. use NLP package to get the word list (after stemming and lemmatization) and the word tags 3. delete the words with pronoun tags and stop words 4. transform words to lower cases. We do these 4 steps for both headline and body text, then concatenate the two list as the article word list. We use different NLP packages for 13 selected languages. So the above pre-processing steps will transform one article (with headline and body) to a list of meaningful words without pronouns and stop words.

For BERT model, since it can learn the context information within sentences, we shouldn't break down sentences into words. The pre-processing steps for BERT model: 1. clean special characters(website addresses, email addresses and etc.) 2. Use NLTK package (sent_tokenize function) to split articles into sentences 3. concatenate the two sentence list of headline and body. Finally, we will get a list of sentences (containing stop words, pronouns and punctuation marks) for each article. Then we use pre-trained base-BERT model to get sentence embeddings and average all sentence embeddings as the article embedings.

3.2.3 Return Prediction and Model Evaluation

We focus on the daily return prediction for all models. We use the three-day-compound return $r_{t-1,t+1}$ as the target variable.

$$r_{t-1,t+1} = (r_{t-1} + 1)(r_t + 1)(r_{t+1} + 1) - 1, \tag{3.1}$$

where r_t is close-to-close return at date t.

When merging returns to each article, the return merging date t is not exactly equal to the article releasing date (local date). If the article releasing date is holiday, then we set return merging date as the next trading date. If the article releasing date is trading day and the releasing timestamp is after market close (e.g. 4 p.m. for US market), we also set return merging date as the next trading date. We set the return merging date as the releasing date only if the releasing date is trading day and the releasing timestamp is before market close.

We evaluation all models with the out-of-sample time-series average of cross-sectional spearman-rank correlation(%) and the performance of daily long-short portfolios. We divide the 24 years data into 10 years of training sample (1996-2005), 5 years of validation sample (2006-2010), and the remaining 9 years for out-of-sample evaluation (2011-2019). We refit the model once a year and implement the model to do out-of-sample prediction in the next year. Each time we refit the model, we will roll over the training sample and validation sample by one year. We maintain the sample size of the training and validation set, but roll them forward to include the most recent year.

To assess out-of-sample predictive performance for 3-day-returns, we calculate the time-series average of cross-sectional spearman-rank correlation(%) between return prediction and day t-1,t,t+1 returns.

$$C_{oos}^{t-1} = \frac{1}{T}\sum_{t=1}^{T} corr(\widehat{r_{t-1,t+1}}, r_{t-1}) \tag{3.2}$$

$$C_{oos}^{t} = \frac{1}{T}\sum_{t=1}^{T} corr(\widehat{r_{t-1,t+1}}, r_{t}) \tag{3.3}$$

$$C_{oos}^{t+1} = \frac{1}{T}\sum_{t=1}^{T} corr(\widehat{r_{t-1,t+1}}, r_{t+1}) \tag{3.4}$$

Based on the stock return prediction, we can build long-short portfolios and evaluate the performance of the portfolios. We follow the strategy in the paper Ke et al. [2019]. At trading day t, we exclude 30 minutes before market opening time (use US market as the illustrated example). We consider all news from trading day t-1 9:00 am to trading day t 9:00 am, and compute the sentiment score for each stock. If no story mentions the stock, the score will be set to 0. If multiple stories mention the stock, the score would be the average of all stories. Among the stock list with sentiment scores, we buy top 50 and sell bottom 50 on the market opening price (if there are less 50 stocks with positive scores, we will buy less). We build positions at the market opening everyday and rebalance at the next market opening, holding the positions of the portfolio within the day. We call this portfolio day+1 portfolio. Similarly, we can define day 0 and day-1 portfolios. The day+1 portfolios are tradable portfolios and day-1 day 0 are not tradable.

3.2.4 Model Comparison

In this section we will compare the performance of different methods. We select 5 major markets US, China (Hong Kong market), UK, Australia and Canada (with sufficient News stories so that the results are more reliable) to report in the main text, and the other minor markets are put in the appendix.

In the table 3.2 and 3.4 we can find that: 1. all 5 major markets and most minor markets have large positive correlation coefficients on day-1 and day 0 and small positive correlation coefficients on day+1, which makes sense because day+1 correlation is tradable but day-1 and day 0 high correlation is forward-looking and not not tradable. The phenomenon also shows that the News dataset contains much useful information related to stock returns. 2. Among major markets, Canada and UK have highest correlation coefficients, followed by US and

Table 3.2: Out-of-Sample Correlation (%) Comparison

Country		BOW	word2vec			bert			bert multilingual		
			LR	LRW	OLS	LR	LRW	OLS	LR	LRW	OLS
US	day-1	3.36	3.74	3.78	3.92	4.73	4.74	5.18	3.08	3.11	4.12
	day 0	6.77	7.60	7.74	8.06	8.73	8.86	9.57	5.56	5.69	6.98
	day+1	1.61	1.65	1.51	1.46	1.63	1.54	1.38	1.46	1.33	1.08
China	day-1	2.27	2.14	1.85	1.82	2.61	2.14	2.29	2.45	2.05	2.22
	day 0	6.67	7.84	7.94	7.57	8.35	8.1	7.71	7.68	7.77	7.61
	day+1	1.61	1.97	1.73	1.45	1.84	1.84	1.17	2.15	1.83	1.3
UK	day-1	5.83	6.10	6.11	5.91	6.51	6.42	6.61	4.76	4.38	5.26
	day 0	10.25	9.98	10.43	11.06	9.33	10.77	11.45	7.16	8.43	9.78
	day+1	2.24	3.76	3.35	2.89	3.59	3.07	2.71	3.08	2.77	3.07
Australia	day-1	5.81	4.97	5.55	5.64	6.32	7.49	7.60	5.47	6.47	6.77
	day 0	5.99	5.65	7.08	7.16	8.44	10.64	10.31	7.14	9.12	9.24
	day+1	0.00	0.30	0.27	-0.22	1.21	0.50	0.09	0.89	0.05	-0.55
Canada	day-1	2.55	4.25	4.35	4.35	4.50	5.14	4.91	3.46	4.08	4.42
	day 0	5.39	5.87	8.29	9.17	6.29	8.86	9.65	3.98	7.64	8.61
	day+1	2.32	5.40	4.51	3.34	5.13	4.44	3.50	4.00	2.82	2.34

Note: In this table, we report the out-of-sample time-series average of cross-sectional spearman-rank correlation(%) between news sentiment prediction with 3-day returns for main countries and all models. bert model is languages-specific model (bert-English and bert-Chinese), and bert multilingual model is for all languages. LR, LRM and OLS indicates logistics regression, return-weighted logistics regression and simple OLS.

China, and Australia News data has lowest prediction power. 3. When comparing different methods, the benchmark BOW method delivers positive correlations and our advanced NLP methods have better performance especially on day+1. And language-specific BERT model dominate BERT multilingual model which is trained using all languages data together. 4. Two logistic regression models are highly correlated between each other and offer us close and robust performance. And logistic regressions are better than OLS regression on day+1 prediction.

Table 3.3: Out-of-Sample Equal-weighted Long-Short Portfolios Comparison

			word2vec			bert			bert multilingual		
Country		BOW	LR	LRW	OLS	LR	LRW	OLS	LR	LRW	OLS
US	Long Ret	10.9	9.4	8.0	7.4	4.9	3.3	2.3	4.3	3.5	2.2
	Long SR	2.28	2.65	2.19	1.94	1.51	0.97	0.60	1.50	1.17	0.66
	Short Ret	10.0	11.1	9.0	10.0	10.0	8.7	10.2	6.6	6.0	5.5
	Short SR	2.41	2.70	2.29	2.50	2.37	2.21	2.44	1.76	1.76	1.57
	L-S Ret	20.9	20.5	17.0	17.5	14.8	12.0	12.5	10.9	9.5	7.8
	L-S SR	3.81	4.67	3.83	3.80	3.44	2.85	2.70	2.94	2.69	1.98
China	Long Ret	-4.5	17.3	13.8	13.8	14.9	15.2	13.7	14.0	14.5	16.2
	Long SR	-0.14	0.86	1.65	1.64	0.75	1.81	1.56	0.70	1.72	1.79
	Short Ret	31.2	4.3	25.2	17.4	2.0	22.1	11.9	2.9	19.7	13.6
	Short SR	1.72	0.58	2.08	1.52	0.26	2.02	1.43	0.37	2.13	1.69
	L-S Ret	26.7	21.7	39.0	31.2	16.9	37.2	25.6	16.9	34.2	29.8
	L-S SR	0.76	1.08	3.12	2.68	0.86	3.26	2.71	0.85	3.35	3.14
UK	Long Ret	14.0	19.0	19.5	19.2	18.0	18.6	16.4	18.1	17.8	17.0
	Long SR	1.84	2.62	2.82	2.56	2.73	2.76	2.39	2.59	2.66	2.49
	Short Ret	-4.0	2.0	9.5	8.1	-1.0	4.9	1.0	0.8	7.6	5.9
	Short SR	-0.58	0.27	1.01	0.90	-0.14	0.60	0.14	0.10	0.85	0.80
	L-S Ret	10.0	21.0	29.0	27.3	17.0	23.5	17.5	18.9	25.4	22.9
	L-S SR	1.47	2.94	3.30	3.05	2.77	3.30	2.71	2.57	3.18	3.48
Australia	Long Ret	-0.5	6.2	6.6	6.1	7.3	11.0	11.0	6.0	9.4	7.4
	Long SR	-0.05	0.66	0.81	0.72	0.86	1.37	1.34	0.71	1.15	0.90
	Short Ret	4.0	-5.6	9.6	6.4	-5.3	3.3	-1.6	-6.5	-0.2	-0.3
	Short SR	0.32	-0.70	0.74	0.53	-0.67	0.32	-0.17	-0.79	-0.02	-0.03
	L-S Ret	3.5	0.7	16.2	12.4	2.0	14.2	9.5	-0.5	9.1	7.2
	L-S SR	0.24	0.07	1.28	1.03	0.24	1.40	1.05	-0.06	0.81	0.75
Canada	Long Ret	31.9	27.1	35.8	37.5	29.7	39.2	41.4	27.4	34.8	36.3
	Long SR	3.64	3.63	4.48	4.34	4.07	4.77	4.59	3.51	4.17	4.06
	Short Ret	-12.4	-18.1	29.1	26.9	-13.6	14.2	13.2	-18.4	7.7	11.6
	Short SR	-1.26	-1.94	2.18	2.34	-1.42	1.29	1.43	-1.95	0.64	1.20
	L-S Ret	19.5	9.0	64.9	64.4	16.2	53.3	54.7	9.0	42.5	47.9
	L-S SR	1.78	0.92	4.68	5.12	1.65	4.42	4.93	0.88	3.22	4.15

Note: In this table, we report the out-of-sample average returns (bps) and sharp ratios of equal-weighted long-short portfolios for main countries and all models. bert model is languages-specific model (bert-English and bert-Chinese), and bert multilingual model is for all languages. LR, LRM and OLS indicates logistics regression, return-weighted logistics regression and simple OLS.

In table 3.3 and 3.5, we report the performance of equal-weighted long-short portfolios for major markets and minor markets. We can see the advanced NLP methods (word2vec, Bert) achieve better performance on the return prediction task, which leads to large economic gain on the performance of long-short portfolios, especially in China, UK and Canada markets. In US market, BOW has a decent Sharp ratio 3.81 then the word2vec method achieves a little better Sharp ratio (4.67 for Logistic regression), and BERT model has a little lower Sharp ratios. In China, UK and Canada markets, BOW's Sharp ratios are around 1, however, the word2vec and BERT both achieve 3+ Sharp ratios in these 3 markets. For Australia market, the sharp ratios of all methods are not significant.

For minor markets, we only report Japan, Germany, Italy, France and Sweden markets because others don't have enough news to trade. The Sharp rations are around 1 or even negative in these markets. The most important reason is that the long-short portfolios only buy and sell around 20 stocks per day (lack of news) and the data quality is not that good (only focus on big companies).

3.2.5 Word Importance

Although we find the News has some prediction power on the stock returns, we may ask what kind of articles/sentences/words will deliver higher/lower returns in the future? To answer this question, we must check the word importance of the model. We also follow the analysis methodology of Bag-of-words to open the black box of word2vec. Because the BERT model gives us different embeddings for the same word (in different context), it is hard to investigate the word importance.

We define the word importance equal to

$$\text{VIP}(word) = \beta \times f_w(word).$$

So the word importance is equal to the article score when the article only has a single word.

We can see the English word importance plots really make sense and consistent with our common knowledge. The most positive words are buy, rise, raise, strong and so on. The most negative words are cut, sell, loss, fall and so on. In the appendix, we also report some other languages word plots.

Figure 3.4: Word Importance Plots

US Positive

US Negative

UK Positive

UK Negative

Figure 3.4: Word Importance Plots (Continued)

Canada Positive

Canada Negative

Australia Positive

Australia Negative

Note: These figures show the most positive English words and the most negative English words in different markets.

3.3 Conclusion

In this paper we study the application of modern NLP methods (word2vec and BERT) to the empirical asset pricing. We work on a huge News dataset covering most of markets and countries all over the world. We find that Natural Language Processing methods significantly improve the stock return prediction performance in many markets. Word2vec and BERT models can extract useful features from the complicated unstructured data and the numerical embeddings can be easily fed to regression models. The return prediction by NLP models also delivers higher Sharp ratios of long-short portfolios.

Word2vec and BERT are promising in this empirical asset pricing task. However, the paper is not completed to some degree. There are still much further research to do after that. For example, how to improve the performance of BERT models and learn the context between articles efficiently? So far our BERT model only can learn the context within a sentence. And how to interpret BERT models and how to visualize the useful context that BERT learned from the articles.

3.4 Appendix

In the appendix, we include the figures which plot the annual time series of the total number of articles for all countries from January 1996 to July 2019. We only provide the total number of articles in 7 months for 2019 (highlighted in red)

Table 3.4: Out-of-Sample Correlation (%) Comparison

Country		BOW	word2vec			bert multilingual		
			LR	LRW	OLS	LR	LRW	OLS
Japan	day-1	28.18	23.73	24.10	24.72	22.41	23.14	24.76
	day 0	22.00	18.35	18.56	18.20	18.39	18.53	20.03
	day+1	-3.06	-0.42	-0.78	-0.82	-0.94	-1.38	-1.63
Germany	day-1	5.33	4.78	3.86	4.31	5.37	6.05	6.13
	day 0	5.91	5.17	5.42	5.89	6.35	7.83	7.53
	day+1	1.04	0.54	1.00	1.01	1.10	1.48	1.46
Italy	day-1	5.08	2.47	2.52	2.55	4.86	4.67	5.58
	day 0	7.74	4.84	5.26	5.18	7.26	7.53	7.34
	day+1	1.22	-0.24	-0.33	-0.32	2.32	1.29	2.47
France	day-1	10.29	9.84	9.13	9.48	6.78	5.81	7.65
	day 0	9.60	9.33	9.50	9.67	8.81	9.52	9.89
	day+1	1.32	2.08	2.69	2.49	0.81	0.92	1.05
Sweden	day-1	3.03	3.45	4.07	4.33	4.05	4.17	4.52
	day 0	6.59	7.53	8.72	9.28	8.13	9.22	9.13
	day+1	1.22	0.88	0.50	-0.37	1.51	1.38	0.01
Denmark	day-1	1.53	1.98	1.69	1.68	2.42	2.96	5.04
	day 0	5.22	6.09	4.16	4.06	2.50	1.83	3.64
	day+1	-0.11	2.74	1.45	1.08	0.88	0.10	1.16
Spain	day-1	2.94	4.94	3.85	2.49	4.06	4.91	4.53
	day 0	8.86	8.23	8.85	9.57	4.42	5.94	4.68
	day+1	2.78	2.67	2.04	1.97	2.49	2.61	3.69
Finland	day-1	-0.83	0.08	0.86	1.87	1.36	1.67	0.94
	day 0	0.41	2.15	2.70	2.57	4.21	3.85	2.65
	day+1	2.33	2.93	1.45	1.38	2.27	1.87	1.88
Greece	day-1	-0.93	-1.44	-0.85	1.53	1.68	-1.08	2.45
	day 0	1.28	1.59	2.14	1.64	2.19	-1.91	-1.24
	day+1	8.47	-2.35	-5.02	-2.77	3.52	4.92	1.54
Portgugal	day-1	8.32	8.83	6.95	8.13	6.22	7.28	5.09
	day 0	12.28	14.38	13.40	13.38	11.48	12.20	14.01
	day+1	1.81	6.57	5.87	6.88	4.66	6.93	5.49
Netherlands	day-1	-5.48	-1.91	1.89	3.16	2.69	2.81	3.16
	day 0	0.64	-7.47	2.90	-1.18	4.01	-1.00	-0.95
	day+1	-5.76	-1.80	1.93	3.71	-2.01	-0.17	1.00

Note: In this table, we report the out-of-sample time-series average of cross-sectional spearman-rank correlation(%) between news sentiment prediction with 3-day returns for other countries and all models. bert model is languages-specific model (bert-English and bert-Chinese), and bert multilingual model is for all languages. LR, LRM and OLS indicates logistics regression, return-weighted logistics regression and simple OLS.

Table 3.5: Out-of-Sample Equal-weighted Long-Short Portfolios Comparison

Country		BOW	word2vec			bert multilingual		
			LR	LRW	OLS	LR	LRW	OLS
Japan	Long Ret	-28.9	-20.7	-21.1	-22.9	-24.3	-25.5	-24.7
	Long SR	-1.74	-1.57	-1.75	-1.86	-1.92	-2.17	-2.09
	Short Ret	12.2	21.1	18.2	19.1	18.3	14.0	14.1
	Short SR	0.91	1.84	1.45	1.55	1.52	1.07	1.14
	L-S Ret	-16.7	0.4	-2.9	-3.8	-6.0	-11.6	-10.6
	L-S SR	-0.85	0.03	-0.20	-0.25	-0.40	-0.77	-0.74
Germany	Long Ret	1.9	2.2	2.5	2.6	2.3	3.7	3.4
	Long SR	0.25	0.34	0.40	0.42	0.35	0.55	0.51
	Short Ret	3.0	2.7	7.9	8.3	2.6	13.4	3.2
	Short SR	0.40	0.25	0.87	0.88	0.30	1.35	0.35
	L-S Ret	5.0	4.8	10.4	10.9	4.9	17.1	6.6
	L-S SR	0.59	0.47	1.09	1.12	0.58	1.67	0.71
Italy	Long Ret	6.2	-2.8	-1.5	-1.7	2.0	-0.7	1.2
	Long SR	0.57	-0.29	-0.19	-0.22	0.23	-0.09	0.14
	Short Ret	3.2	2.0	2.7	1.2	5.3	3.3	6.3
	Short SR	0.27	0.24	0.28	0.13	0.55	0.33	0.67
	L-S Ret	9.4	-0.8	1.2	-0.5	7.2	2.6	7.5
	L-S SR	0.66	-0.07	0.12	-0.05	0.72	0.25	0.78
France	Long Ret	1.7	2.3	3.4	3.8	3.0	2.6	3.7
	Long SR	0.22	0.34	0.52	0.57	0.45	0.40	0.52
	Short Ret	9.6	4.2	9.0	9.1	2.7	2.4	4.2
	Short SR	1.00	0.47	1.02	1.08	0.31	0.23	0.55
	L-S Ret	11.3	6.5	12.3	12.9	5.7	5.1	8.0
	L-S SR	1.10	0.74	1.39	1.53	0.66	0.48	0.96
Sweden	Long Ret	2.0	-5.1	-4.6	-5.2	-1.0	-3.4	-3.5
	Long SR	0.18	-0.56	-0.54	-0.61	-0.11	-0.39	-0.39
	Short Ret	16.5	10.9	10.5	6.4	12.7	14.5	11.5
	Short SR	1.17	0.82	0.62	0.41	0.97	0.89	0.84
	L-S Ret	18.5	5.8	5.9	1.2	11.7	11.1	8.0
	L-S SR	1.12	0.41	0.33	0.07	0.82	0.65	0.55

Note: In this table, we report the out-of-sample average returns (bps) and sharp ratios of equal-weighted long-short portfolios for other countries and all models. bert model is languages-specific model (bert-English and bert-Chinese), and bert multilingual model is for all languages. LR, LRM and OLS indicates logistics regression, return-weighted logistics regression and simple OLS.

Figure 3.5: Average Article Counts By Country

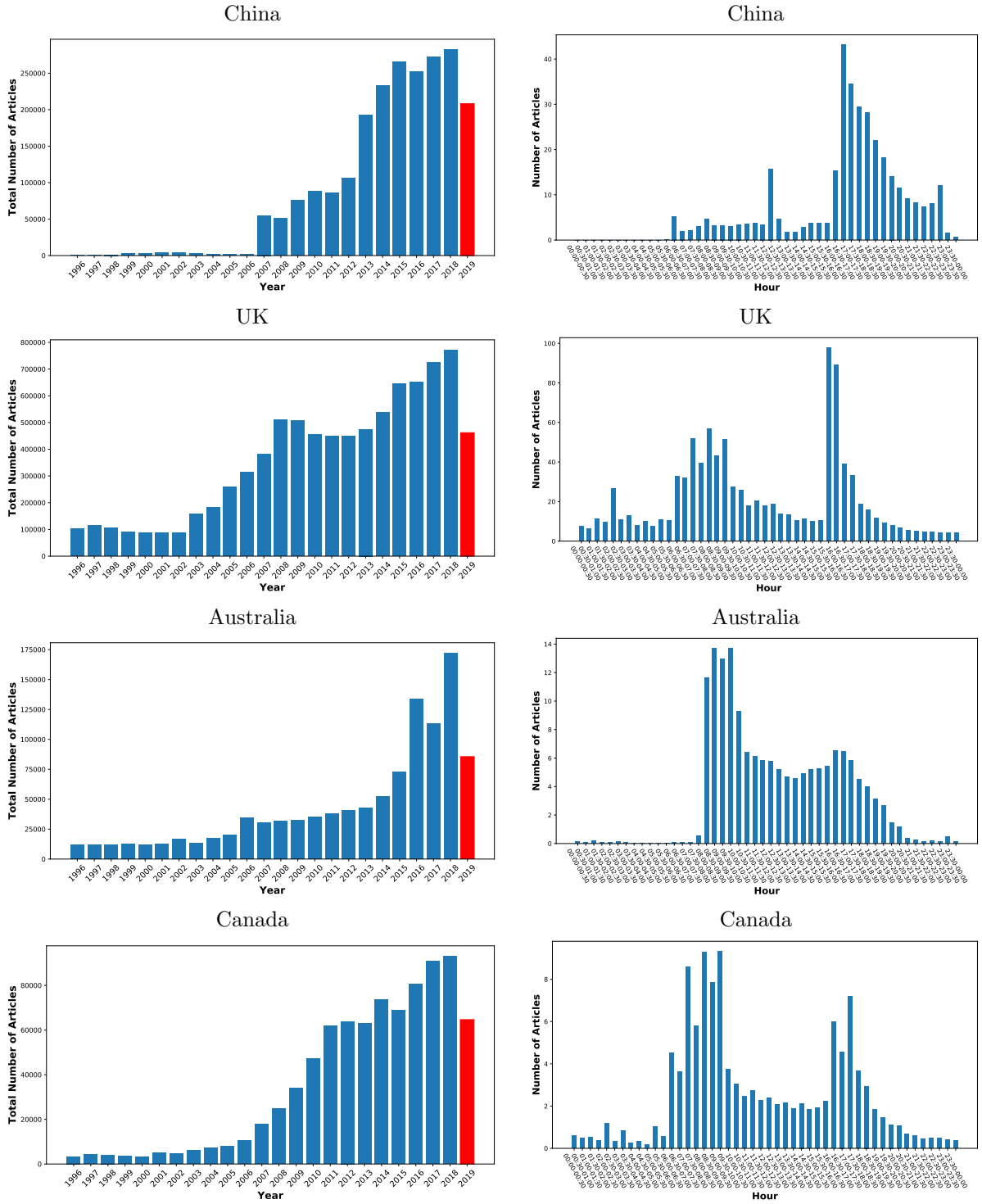


Figure 3.5: Average Article Counts By Country (Continued)

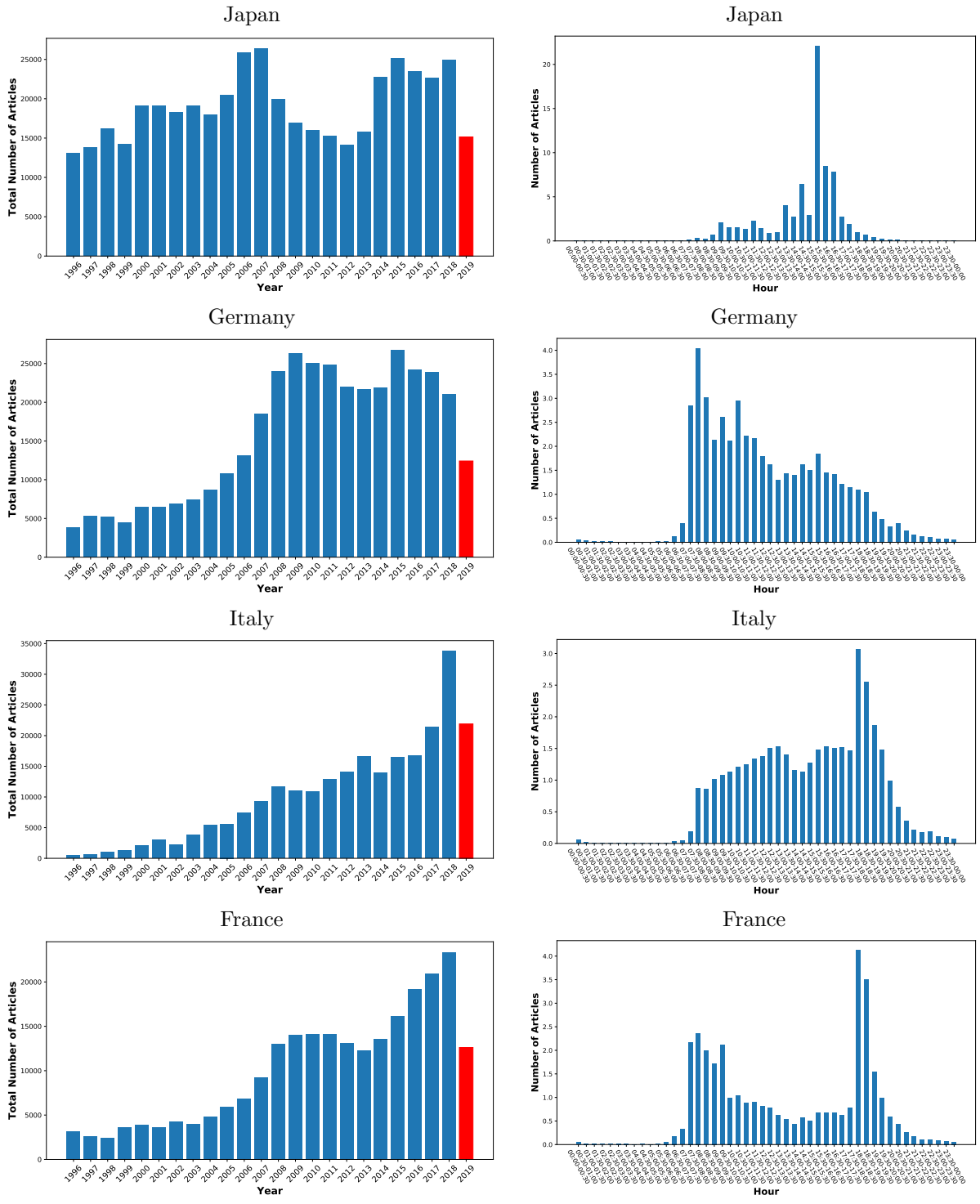


Figure 3.5: Average Article Counts By Country (Continued)

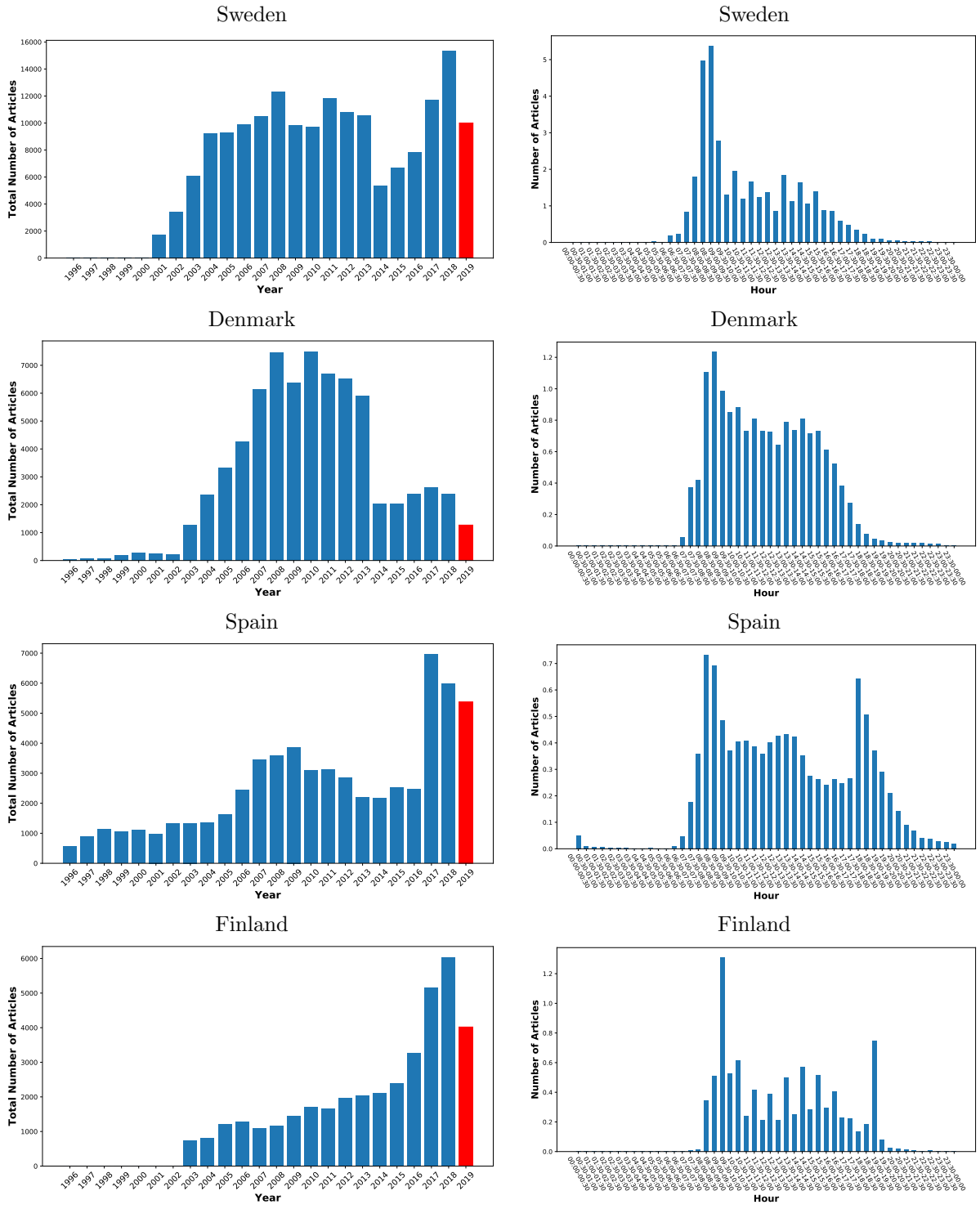
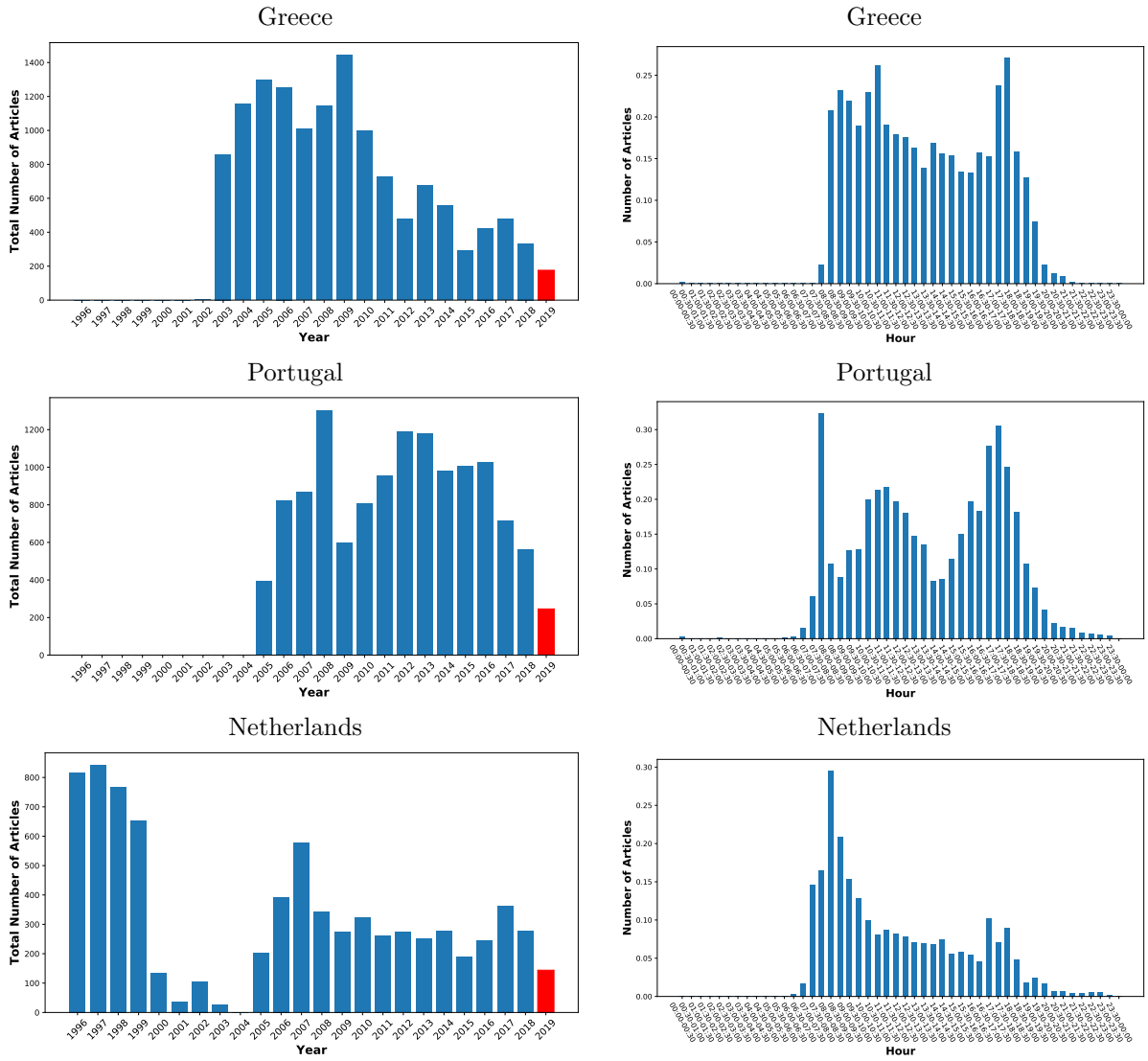


Figure 3.5: Average Article Counts By Country (Continued)



Note: This left figures plot the annual time series of the total number of articles for all countries from January 1996 to July 2019. We only provide the total number of articles in 7 months for 2019 (highlighted in red). The right figures plot the average numbers of articles per hour (local trading time).

Figure 3.6: Word Importance Plots

France Positive

France Negative

Germany Positive

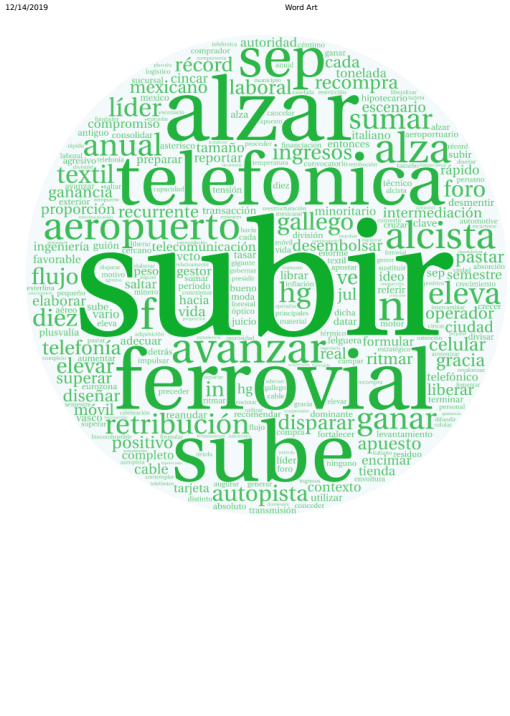
Germany Negative

Figure 3.6: Word Importance Plots (Continued)

Italy Positive

Italy Negative

Spain Positive



Spain Negative

REFERENCES

Seung C. Ahn and Alex R. Horenstein. Eigenvalue ratio test for the number of factors. *Econometrica*, 81:1203–1227, 2013.

Yacine Aït-Sahalia and Dacheng Xiu. Using principal component analysis to estimate a high dimensional factor model with high-frequency data. *Journal of Econometrics*, 201:388–399, 2017.

Lucia Alessi, Matteo Barigozzi, and Marco Capasso. Improved penalization for determining the number of factors in approximate factor models. *Statistics and Probability Letters*, 80: 1806–1813, 2010.

Dante Amengual and Mark W. Watson. Consistent estimation of the number of dynamic factors in a large N and T panel. *Journal of Business and Economic Statistics*, 25:91–96, 2007.

Francis R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, June 2008. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1390681.1390721`.

Jushan Bai. Inferential Theory for Factor Models of Large Dimensions. *Econometrica*, 71(1): 135–171, 2003. ISSN 0012-9682. doi: 10.1111/1468-0262.00392.

Jushan Bai and Serena Ng. Determining the number of factors in approximate factor models. *Econometrica*, 70:191–221, 2002.

Jushan Bai and Serena Ng. Principal components estimation and identification of static factors. *Journal of Econometrics*, 176(1):18–29, September 2013.

Jushan Bai and Serena Ng. Principal components and regularized estimation of factor models. Technical report, Columbia University, 2017.

Zhidong Bai. Methodologies in spectral analysis of large dimensional random matrices: A review. *Statistica Sinica*, 9:611–677, 1999.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Ravi Bansal and Amir Yaron. Risks for the long run: A potential resolution of asset pricing puzzles. *The journal of Finance*, 59(4):1481–1509, 2004.

Gérard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13: 1063–1095, 2012.

Peter J Bickel, Ya'acov Ritov, and Alexandre B Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of Statistics*, 37(4):1705–1732, August 2009.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

G. E. P. Box. Non-normality and tests on variances. *Biometrika*, 40:318–335, 1953.

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

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Peter Bühlmann and Torsten Hothorn. Boosting Algorithms: Regularization, Prediction and Model Fitting. *Statistical Science*, 22(4):477–505, November 2007.

Peter Bühlmann and Bin Yu. Boosting with the l2 loss. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

Florentin Butaru, Qingqing Chen, Brian Clark, Sanmay Das, Andrew W Lo, and Akhtar Siddique. Risk and risk management in the credit card industry. *Journal of Banking & Finance*, 72:218–239, 2016.

John Y Campbell and John H Cochrane. By force of habit: A consumption-based explanation of aggregate stock market behavior. *Journal of political Economy*, 107(2):205–251, 1999.

John Y. Campbell and S. B. Thompson. Predicting excess stock returns out of sample: Can anyting beat the historical average? *Review of Financial Studies*, 21(4):1509–1531, 2008.

John Y Campbell and Samuel B Thompson. Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies*, 21(4):1509–1531, 2007.

Gary Chamberlain and Michael Rothschild. Arbitrage, factor structure, and mean-variance analysis on large asset markets. *Econometrica*, 51:1281–1304, 1983.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 3040–3050, USA, 2018. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=3327144.3327226`.

John H Cochrane. The dog that did not bark: A defense of return predictability. *The Review of Financial Studies*, 21(4):1533–1575, 2007.

Gregory Connor and Robert A Korajczyk. Performance measurement with the arbitrage pricing theory: A new framework for analysis. *Journal of Financial Economics*, 15(3): 373–394, 1986.

Gregory Connor, Matthias Hagmann, and Oliver Linton. Efficient semiparametric estimation of the fama–french model and extensions. *Econometrica*, 80(2):713–754, 2012.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

I Daubechies, M Defrise, and C De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, November 2004.

Sijmen de Jong. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18(3):251 – 263, 1993. ISSN 0169-7439. doi: https://doi.org/10.1016/0169-7439(93)85002-X. URL `http://www.sciencedirect.com/science/article/pii/016974399385002X`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Francis X. Diebold. 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):1–9, 2015.

Francis X. Diebold and Roberto S. Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):134–144, 1995.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Yannis Dimopoulos, Paul Bourret, and Sovan Lek. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, 2(6):1–4, 1995.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v49/eldan16.html`.

Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56, 1993.

Eugene F Fama and Kenneth R French. Dissecting anomalies. *The Journal of Finance*, 63 (4):1653–1678, 2008.

Eugene F. Fama and Kenneth R. French. A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22, 2015. ISSN 0304405X. doi: 10.1016/j.jfineco.2014.10.010. URL http://dx.doi.org/10.1016/j.jfineco.2014.10.010.

Jianqing Fan, Yuan Liao, and Weichen Wang. Projected principal component analysis in factor models. *Annals of Statistics*, 44(1):219, 2016.

Jianqing Fan, Quefeng Li, and Yuyan Wang. Estimation of high dimensional mean regression in the absence of symmetry and light tail assumptions. *Journal of the Royal Statistical Society, B*, 79(1):247–265, 2017.

Jianqing Fan, Cong Ma, and Yiqiao Zhong. A selective overview of deep learning. Technical report, Princeton University, 2019.

Guanhao Feng, Stefano Giglio, and Dacheng Xiu. Taming the factor zoo: A test of new factors. *Journal of Finance, forthcoming*, 2019a.

Guanhao Feng, Nicholas G. Polson, and Jianeng Xu. Deep learning in asset pricing. Technical report, University of Chicago, 2019b.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, 1995.

Joachim Freyberger, Andreas Neuhierl, and Michael Weber. Dissecting characteristics nonparametrically. *Review of Financial Studies, forthcoming*, 2019.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000a.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of Statistics*, 28(2):337–407, 04 2000b. doi: 10.1214/aos/1016218223. URL https://doi.org/10.1214/aos/1016218223.

Jerome Friedman, Trevor Hastie, Holger Höfling, Robert Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Patrick Gagliardini, Elisa Ossola, and Olivier Scaillet. Time-varying risk premium in large cross-sectional equity datasets. *Econometrica*, 84(3):985–1046, 2016.

Patrick Gallinari, Yann LeCun, Sylvie Thiria, and Francoise Fogelman-Soulie. Memoires associatives distribuees. *Proceedings of COGNITIVA*, 87:93, 1987.

Stefano W Giglio and Dacheng Xiu. Asset pricing with omitted factors. Technical report, University of Chicago, 2016.

Stefano W Giglio and Dacheng Xiu. Asset pricing with omitted factors. Technical report, University of Chicago, 2018.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, pages 315–323, 2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Jeremiah Green, John RM Hand, and X Frank Zhang. The supraview of return predictive signals. *Review of Accounting Studies*, 18(3):692–730, 2013.

Jeremiah Green, John RM Hand, and X Frank Zhang. The characteristics that provide independent information about average us monthly stock returns. *The Review of Financial Studies*, 30(12):4389–4436, 2017.

Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *Review of Financial Studies, forthcoming*, 2019a.

Shihao Gu, Bryan T Kelly, and Dacheng Xiu. Autoencoder asset pricing models. *Available at SSRN*, 2019b.

Marc Hallin and Roman Liška. Determining the number of factors in the general dynamic factor model. *Journal of the American Statistical Association*, 102(478):603–617, June 2007.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

Campbell R. Harvey and Wayne E. Ferson. Conditioning variables and the cross-section of stock returns. *Journal of Finance*, 54:1325–1360, 1999.

Campbell R Harvey and Yan Liu. Lucky factors. Technical report, Duke University, 2016.

Campbell R Harvey, Yan Liu, and Heqing Zhu. ... and the cross-section of expected returns. *Review of Financial Studies*, 29(1):5–68, 2016.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016a. doi: 10.1109/CVPR.2016.90. URL `https://doi.org/10.1109/CVPR.2016.90`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016b.

Zhiguo He and Arvind Krishnamurthy. Intermediary asset pricing. *American Economic Review*, 103(2):732–70, 2013.

JB Heaton, NG Polson, and JH Witte. Deep learning in finance. *arXiv preprint arXiv:1602.06561*, 2016.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10, 1994.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL `http://dx.doi.org/10.1162/neco.2006.18.7.1527`.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49 (3):851–889, 1994.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*, pages 448–456, June 2015.

Martin Jaggi. An equivalence between the lasso and support vector machines. *Regularization, optimization, kernels, and support vector machines*, pages 1–26, 2013.

Kevin Jarrett, Koray Kavukcuoglu, Yann Lecun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. IEEE, 2009.

Iain M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of Statistics*, 29:295–327, 2001.

Iain M. Johnstone and Arthur Yu Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486): 682–693, 2009.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016a.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016b.

George Kapetanios. A testing procedure for determining the number of factors in approximate factor models. *Journal of Business and Economic Statistics*, 28:397–409, 2010.

Zheng Tracy Ke, Bryan T Kelly, and Dacheng Xiu. Predicting returns with text data. Technical report, National Bureau of Economic Research, 2019.

Bryan Kelly and Seth Pruitt. Market expectations in the cross-section of present values. *The Journal of Finance*, 68(5):1721–1756, 2013.

Bryan Kelly and Seth Pruitt. The three-pass regression filter: A new approach to forecasting using many predictors. *Journal of Econometrics*, 186(2):294–316, 2015.

Bryan Kelly, Seth Pruitt, and Yinan Su. Characteristics *are* covariances: A unified model of risk and return. *Journal of Financial Economics*, 2019.

Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Keith Knight and Wenjiang Fu. Asymptotics for lasso-type estimators. *Annals of Statistics*, 28 (5):1356–1378, 10 2000. doi: 10.1214/aos/1015957397. URL https://doi.org/10.1214/aos/1015957397.

Ralph Koijen and Stijn Van Nieuwerburgh. Predictability of returns and cash flows. *Annual Review of Financial Economics*, 3:467–491, 2011.

Serhiy Kozak. Kernel trick for the cross section. *Available at SSRN 3307895*, 2019.

Serhiy Kozak, Stefan Nagel, and Shrihari Santosh. Interpreting factor models. *Journal of Finance*, 73(3):1183–1223, 2018.

Serhiy Kozak, Stefan Nagel, and Shrihari Santosh. Shrinking the cross section. *Journal of Financial Economics, forthcoming*, 2019.

Jonathan Lewellen. The cross-section of expected stock returns. *Critical Finance Review*, 4 (1):1–44, 2015.

Henry W. Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, Sep 2017. ISSN 1572-9613. doi: 10.1007/s10955-017-1836-5. URL https://doi.org/10.1007/s10955-017-1836-5.

Andrew W Lo and A Craig MacKinlay. Data-snooping biases in tests of financial asset pricing models. *Review of financial studies*, 3(3):431–467, 1990.

Karim Lounici, Massimiliano Pontil, Sara van de Geer, and Alexandre B. Tsybakov. Oracle inequalities and optimal inference under group sparsity. *Annals of Statistics*, 39(4):2164–2204, 08 2011. doi: 10.1214/11-AOS896. URL https://doi.org/10.1214/11-AOS896.

Gábor Lugosi and Nicolas Vayatis. On the bayes-risk consistency of regularized boosting methods. *Annals of Statistics*, 32(1):30–55, 02 2004. doi: 10.1214/aos/1079120129. URL https://doi.org/10.1214/aos/1079120129.

Timothy Masters. *Practical Neural Network Recipes in C++*. Academic Press, 1993.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1806579115. URL `https://www.pnas.org/content/115/33/E7665`.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory (COLT)*, 2019.

Nicolai Meinshausen and Bin Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 37(1):246–270, 2009.

Lucas Mentch and Giles Hooker. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research*, 17(26):1–41, 2016. URL `http://jmlr.org/papers/v17/14-168.html`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Christine De Mol, Ernesto De Vito, and Lorenzo Rosasco. Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201 – 230, 2009. ISSN 0885-064X. doi: https://doi.org/10.1016/j.jco.2009.01.002. URL `http://www.sciencedirect.com/science/article/pii/S0885064X0900003X`.

Hyungsik Roger Moon and Martin Weidner. Nuclear norm regularized estimation of panel regression models. Technical report, University of Southern California, 2018.

Benjamin Moritz and Tom Zimmermann. Tree-based conditional portfolio sorts: The relation between past and future stock returns. *Available at SSRN 2740751*, 2016.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Alexei Onatski. Determining the number of factors from empirical distribution of eigenvalues. *Review of Economics and Statistics*, 92:1004–1016, 2010.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

Debashis Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistical Sinica*, 17:1617–1642, 2007.

Walter Pohl, Karl Schmedders, and Ole Wilms. Higher order effects in asset pricing models with long-run risks. *The Journal of Finance*, 73(3):1061–1111, 2018.

Nicholas G. Polson, James Scott, and Brandon T. Willard. Proximal algorithms in statistics and machine learning. *Statistical Science*, 30(4):559–581, 2015.

David Rapach and Guofu Zhou. Forecasting stock returns. vol. 2 of handbook of economic forecasting, 328 383, 2013.

David E Rapach, Jack K Strauss, and Guofu Zhou. International stock return predictability: what is the role of the united states? *The Journal of Finance*, 68(4):1633–1662, 2013.

Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse additive models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 71(5):1009–1030, 2009.

David Rolnick and Max Tegmark. The power of deeper networks for expressing natural functions. In *ICLR*, 2018.

Barr Rosenberg. Extra-market components of covariance in security returns. *Journal of Financial and Quantitative Analysis*, 9(2):263–274, 1974.

Stephen A Ross. The arbitrage theory of capital asset pricing. *Journal of economic theory*, 13(3):341–360, 1976.

Tano Santos and Pietro Veronesi. Conditional betas. Technical report, National Bureau of Economic Research, 2004.

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

Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *Annals of Statistics*, 43(4):1716–1741, 08 2015. doi: 10.1214/15-AOS1321. URL `https://doi.org/10.1214/15-AOS1321`.

Justin Sirignano, Apaar Sadhwani, and Kay Giesecke. Deep learning for mortgage risk. *Available at SSRN 2799443*, 2016.

James H Stock and Mark W Watson. Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics*, 20(2):147–162, 2002a.

James H. Stock and Mark W. Watson. Forecasting using principal components from a large number of predictors. *Journal of American Statistical Association*, 97:1167–1179, 2002b.

Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.

J. W. Tukey. A survey of sampling from contaminated distributions. In I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, editors, *Contributions to probability and statistics: Essays in Honor of Harold Hotelling*. Stanford University Press, 1960.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *Journal of Machine Learning Research*, 15: 1625–1651, 2014. URL http://jmlr.org/papers/v15/wager14a.html.

Martin J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using l_1-constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, 2009.

Weichen Wang and Jianqing Fan. Asymptotics of empirical eigenstructure for high dimensional spiked covariance. *Annals of Statistics*, 45:1342–1374, 2017.

Ivo Welch and Amit Goyal. A Comprehensive Look at The Empirical Performance of Equity Premium Prediction. *Review of Financial Studies*, 21(4):1455–1508, August 2008.

Kenneth D. West. Forecast evaluation. In Graham Elliott, Cliver Granger, and Allan Timmermann, editors, *Handbook of Economic Forecasting*, volume 1, pages 99–134. Elsevier, 2006.

Halbert White. Using least squares to approximate unknown regression functions. *International Economic Review*, pages 149–170, 1980.

D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural networks*, 16:1429–1451, 2003.

Jingtao Yao, Yili Li, and Chew Lim Tan. Option price forecasting using neural networks. *Omega*, 28(4):455–466, 2000.

Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594, 08 2008. doi: 10.1214/07-AOS520. URL http://dx.doi.org/10.1214/07-AOS520.

Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33(4):1538–1579, 08 2005. doi: 10.1214/009053605000000255. URL https://doi.org/10.1214/009053605000000255.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3647580.

Hui Zou and Hao Helen Zhang. On the adaptive elastic-net with a diverging number of parameters. *Annals of Statistics*, 37(4):1733–1751, 08 2009. doi: 10.1214/08-AOS625. URL https://doi.org/10.1214/08-AOS625.