Overhyped? Can ML Models Reliably Predict Stock Returns?*

S. Yanki Kalfa[1], Allan Timmermann[1], and Terri van der Zwan[2,3]

[1] *University of California San Diego, Rady School of Management*
[2] *Erasmus School of Economics, Erasmus University Rotterdam*
[3] *Tinbergen Institute*

March 27, 2025

Abstract

Hyperparameters determine the architecture of machine learning (ML) models and can greatly affect their forecasting performance, yet there is little consensus on how to choose the range and grid of hyperparameters to search over. We provide an extensive examination of which hyperparameters are most important for popular ML models' out-of-sample forecasting performance using a large U.S. dataset on individual stock returns and firm characteristics. We find that some choices of hyperparameters virtually guarantee good out-of-sample return forecasts while others lock in poor forecasts. This poses a challenge because many empirical studies fail to provide details on how they set their hyperparameters. We also find that time-series validation methods do not offer a definitive solution to the dependence of out-of-sample return forecasting performance on the underlying range of hyperparameters.

*Corresponding email: skalfa@ucsd.edu, atimmermann@ucsd.edu, t.vanderzwan@ese.eur.nl. This paper has greatly benefited from comments from seminar and conference participants at CIREQ-CMP (University of Montreal), Birkbeck College (University of London), USC, Hong Kong University Business School, and Robeco Asset Management.

1 Introduction

Machine learning (ML) methods are now widely used in empirical asset pricing, with numerous studies reporting improved return forecasts and portfolio performance relative to traditional linear models.[1] While early empirical results have been promising, evaluating ML forecasts presents new challenges due to greater model flexibility and complexity of these models. In particular, the forecasting performance of ML methods such as random forests, boosted regression trees and neural nets depends on a high-dimensional set of hyperparameters that control their ability model nonlinearities and interactions. In contrast, conventional (linear) models use regularization to select predictors, requiting relatively simple and well-understood decisions about penalization and shrinkage.

The complexity of ML hyperparameters is especially evident in neural networks (NNS). Researcher must decide on the number of hidden layers and their architecture, activation functions (e.g., linear or ReLU), and a loss function (e.g., MSE or Huber). Because NNs update the input weights iteratively (in epochs), the choice of optimization algorithm and learning rates are also critical. Additional decisions include whether to use batch normalization, dropout rates, early stopping, and regularization in hidden layers.[2]

Given the large number of hyperparameter choices, how do researchers determine the range of values to consider in practice? Table 1 summarizes the hyperparameter settings reported in recent studies using ML methods. Random Forests (RF), Gradient Boosted Regression Trees (GBRT), and Neural Networks (NN) are the most popular algorithms. The table illustrates that there is no broad consensus on the hyperparameter values across studies. This variation reflects the absence of economic guidance, making it difficult to justify universally accepted tuning ranges.[3] Some hyperparameters are simply set ex ante, appealing to prior studies or "convention", while others are tuned through validation methods. Both strategies require an understanding of how the hyperparameters and their interactions

[1]Prominent studies that use ML methods to predict stock returns include Gu et al. (2020), Gu et al. (2021), Jiang et al. (2023), and Han et al. (2023). Kelly et al. (2023) provide a comprehensive recent review of the literature.

[2]This list of hyperparameters and architectural choices is not comprehensive but represents some of the main desiderata researchers must consider when training neural networks.

[3]For example, the learning rate of the Adam optimization algorithm is difficult to choose based on economic intuition or insights.

forecasting performance.

Because an exhaustive search over all hyperparameter combinations is infeasible, researchers limit their search to a subset of the high-dimensional space of hyperparameters. Although hyperparameters play a central role in ML models, their impact on return predictability performance, particularly with respect to their ranges and grid specifications, has received little attention. Most existing studies report specific combinations of hyperparameters that yield strong forecasting performance, without analyzing the sensitivity of results to hyperparameter choice. This paper addresses this gap by systematically analyzing the sensitivity of return forecast performance to hyperparameter settings. We examine the implications of this sensitivity for inference on out-of-sample forecasting performance and explore potential remedies in the form of validation schemes and priors on hyperparameters.[4]

Our empirical analysis uses the updated stock-level characteristics dataset of Gu et al. (2020), merged with monthly stock returns from the Center for Research in Security Prices (CRSP). To address missing values in characteristics, we apply the local backward B-XS model of Bryzgalova et al. (2024) to impute missing characteristics based on historical data and contemporaneous cross-sectional asset information, avoiding look-ahead bias. The sample spans 45 years and includes nearly 7,000 unique firms and 140 characteristics.

To evaluate the sensitivity of out-of-sample forecasting performance to model hyperparameters, we conduct a series of experiments in which one hyperparameter is fixed while the others are allowed to vary freely. For penalized linear models such as the Lasso and Elastic Net, forecasting performance is primarily determined by the shrinkage parameter, which determines the inclusion threshold for covariates. For Random Forests, tree depth and the number of features considered at each split are the most important hyperparameters. In Extreme Gradient Boosted Trees (XGBoost), the key drivers of performance are again tree depth and the learning rate.[5] In neural networks, out-of-sample accuracy is most sensitive to the learning rate and kernel l_1 regularization. In contrast, the number of hidden layers, loss function, and activation function have relatively small effects.

[4]This is also related to the literature on nonstandard errors (see, e.g., Menkveld et al., 2024), which examines the effects of research design choices on results (e.g., explanatory variable transformation, dependent variable selection). We leave this aspect for future research.

[5]The number of boosting rounds has comparatively limited influence.

Building on these insights, we demonstrate that the range of values considered for hyperparameters have a substantial impact on out-of-sample forecasting performance. For instance, setting the shrinkage parameter too low leads to the inclusion of too many predictors, yielding forecasts dominated by noise and poor out-of-sample performance. In the case of XGBoost, shallow trees combined with low learning rate produce overly conservative models that underfit the data, whereas deep trees with high learning rates lead to overfitting and instability during training. Similarly, a high learning rate in neural networks results in significantly lower out-of-sample accuracy on average and greater likelihood of under-performance relative to models with lower learning rates.

Several key insights emerge from this analysis. Across all ML methods, we find that fixing a subset ("corner") of the hyperparameter space is sufficient to achieve good out-of-sample forecasting performance, regardless of the values chosen for the remaining hyperparameters. Conversely, constraining the hyperparameter space in particular ways can lead to poor out-of-sample performance with high probability. This situation typically arises when flexible models are combined with high learning rates.

We further show that out-of-sample predictive performance exhibits considerably wider dispersion for certain machine learning models, such as neural networks, than for traditional linear models. This wider spread increases the risk that the forecasting performance of these methods is overstated ("hyped"), as it is possible to select hyperparameters that yield seemingly strong out-of-sample results.[6] Importantly, the out-of-sample performance of machine learning models is not only more dispersed than that of linear models, but also exhibits a pronounced left-skew. In other words, extremely poor performance is possible under certain hyperparameter settings, while the upside remains limited across a wide range of configurations. In extreme cases, poorly tuned models, especially those with forecasts that are highly sensitive to noise, can perform arbitrarily poor.

The patterns in average forecasting performance of machine learning models also extend to cross-sectional measures of return predictability. Forecast accuracy exhibits greater cross-sectional dispersion for flexible machine learning models relative to penalized linear models.

[6]This phenomenon is distinct from over-fitting, which refers to a model's ability to provide an arbitrarily good (in-sample) fit on the training data. In contrast, we show that machine learning models exhibit much larger dispersion in out-of-sample performance relative to conventional linear forecasting models.

Out-of-sample performance is particularly weak for large-cap stocks and stocks with high trading volume. These results raise concerns about the feasibility of implementing such forecasts in profitable strategies after accounting for transaction costs.

Building these findings, we next examine whether validation methods offer a solution solution for selecting hyperparameters. In empirical applications, researchers partition the data into training, validation, and test samples, using the validation set to guide hyperparameter tuning. This process introduces several additional design choices, including the validation method, the length of the validation window, the specification of the hyperparameter grid, the search algorithm, and use of early-stopping.[7]

We find that these choices materially influence out-of-sample forecasting performance. For instance, the performance of XGBoost and neural networks is highly sensitive to the initial hyperparameter grid. This sensitivity persists even when using adaptive grid search algorithms that refine the search space to improve the likelihood of identifying (near-)optimal configurations. In practice, validation is therefore not a panacea for hyperparameter selection.

Economic reasoning provides little guidance on the appropriate range of hyperparameters for machine learning models. However, the architecture of these models, shaped by hyperparameters, has a direct effect on forecasting performance. Understanding this relationship can inform practical strategies for selecting reasonable tuning ranges. In particular and help regulate the bias-variance trade-off, which is essential to out-of-sample performance.

The remainder of the paper proceeds as follows. Section 2 describes the ML methods we consider, emphasizing the hyperparameters each method uses. Section 3 introduces the main features of our data along with the estimation and forecasting procedure. Section 4 provides a detailed analysis of the ML models' out-of-sample forecasting performance for different regions of the space of hyperparameters. Section 5 examines the forecasting performance under different validation schemes used to select the hyperparameters. Section 6 introduces efficient validation algorithms and priors on hyperparameter ranges. Finally, Section 7 concludes.

[7]Common search algorithms include grid search, randomized search, and Bayesian optimization methods such as Tree-structured Parzen Estimators (TPE).

Table 1: Hyperparameter grids used by previous studies

RF refers to random forest, GBT to gradient boosted trees, NN to neural networks, and CV to cross-validation. "NA" indicates use of the algorithm but no mention of the hyperparameter grid. "–" indicates no use of the algorithm. For the linear models, α corresponds to the l_1 regularization penalty, and λ to the (l_1, l_2) penalty mixer. Further, l_1 and l_2 refer to regularization terms, η denotes the learning rate associated with the algorithm used. For cross-validation, TS refers to time-series cross-validation, POOS refers to pseudo-out-of-sample. * The authors consider multiple datasets with various hyperparameter settings. Here, we present the hyperparameters used in their analysis of the cross-section of expected returns.

	Lasso	Enet	RF	GBT	NN	CV
Gu et al. (2020)	–	$\alpha \in [0.0001, 0.1]$ $\lambda = 0.5$	Trees=300 Max. Depth $\in \{1, ..., 6\}$ Features=3,5,10,20,...	Trees $\in \{1, ..., 1000\}$ Max. Depth $\in \{1, 2\}$ $\eta \in \{0.01, 0.1\}$	Layers $\in \{1, ..., 5\}$ Nodes=[32,16,8,2,1] Adam $\eta \in \{0.001, 0.01\}$ $l_1 \in (10^{-5}, 10^{-3})$ Batch size=10000 Epochs=100 Patience=5 Ensemble=10	TS Validation 10 years
Wolff and Neugebauer (2019)	NA	NA	NA	–	–	5-Fold
Masini et al. (2023)	NA	NA	NA	NA	Layers=1,3,5 Nodes=NA Dropout=NA	NA
Van Binsbergen et al. (2023)	–	–	Trees=2000 Max. Depth=7 Feature Fraction=0.01 Min. Node=5	–	–	NA
Coulombe et al. (2023)	–	–	–	Trees=100 Max. Depth=NA Feature Fraction=NA l_1/l_2=NA Subsample Ratio=NA	–	TS Validation 3 years Optuna "Wide grid"
Bianchi et al. (2021)	–	–	–	–	Layers=1,2 Nodes=[1, 3] Dropout=$\{0.1, 0.3, 0.5\}$ $l_1, l_2 \in \{0.0001, 0.001, 0.01\}$	TS Validation 5 years
Chen et al. (2024)	–	NA	–	–	Layers=3 Nodes=$[32, 16, 8]$ Adam η=0.001 Dropout=0.05	TS Validation 5 years
Avramov et al. (2023)	–	–	–	–	Layers=3 Nodes=[32,16,8] l_1=NA Adam η=NA Batch-normalization	3-Fold 12 years
Coulombe et al. (2022)	–	–	–	Max. Depth=10 Feature Fraction=1	Layers=1,2 Nodes=[32,16] Adam $\eta = 0.01$ Patience=5	POOS and 5-Fold
Bali et al. (2023)	$\alpha \in [10^{-6}, 10^{-2}]$	$\alpha \in [10^{-6}, 10^{-2}]$ $\lambda \in [0, 1]$	Trees=1024 Max. Depth $\in \{2, 10\}$ Max. Leaves $\in [2, 10]$ Feature Fraction $\in [0.25, 1]$	Trees=1024 Max. Depth $\in \{2, 10\}$ Max. Leaves $\in [2, 10]$ Feature Fraction $\in [0.25, 1]$ $\eta \in \{0.01, 0.1, 1\}$ $l_1, l_2 \in [0, 0.1]$	Layers $\in \{1, ..., 5\}$ Batch size $\in \{2^{12}, 2^{14}, 2^{16}\}$ $\eta \in \{0.001, 0.01, 0.1\}$ Dropout $\in [0, 0.5]$ Epochs=64	TS Validation 2 years Enhanced Hyperband
Cao et al. (2024)	–	–	Trees=100 Max. Depth=3	Trees=100 Max. Depth=3 $\eta = 0.05$ Subsample Ratio=0.4	Layers=5 Batch size=1000 $\eta = 0.05$ Epochs=50	NA
Shen and Xiu (2024)*	$\alpha \in [10^{-3.4}, 10^{-2.4}]$	–	Trees=500 Max. Depth $\in \{1, ..., 6\}$ Features $\in \{3, 5, 10, 15, 20\}$	Trees $\in \{1, ..., 10\}$ Max. Depth $\in \{1, ..., 5\}$ $\eta \in \{0.0001, 0.001,$ $0.01, 0.02, 0.05\}$	Layers=1, Nodes=32 Batch size=10000 $(\eta, \text{epochs}) \in$ $\{(0.5, 2), (0.1, 10), (0.067, 15),$ $(0.05, 20), (0.04, 25)\}$ $l_1 \in [10^{-5}, 10^{-3}]$, $l_2 \in [0.0001, 1]$	2-Fold

2 Methodology

This section develops the return prediction framework used in our analysis and introduces the machine learning models used to examine how hyperparameter choices affect forecasting per-

formance. We begin by defining a flexible function, mapping firm characteristics to expected returns. We then describe three classes of models: penalized regressions, tree based models, and feed-forward neural networks, each representing different level of complexity. For each model class, we outline its structure, associated hyperparameters and their configurations used in our analysis.

2.1 The Modeling Problem

Let $r_{i,t}$ denote the excess returns of stock i at time t, where $i = 1, \ldots, N_t$ and $t = 1, \ldots, T$. We add a t subscript to N to note the unbalanced structure of the panel. However, to simplify notation, we refer to the number of stocks in each period as N in the remainder.

We decompose excess returns into their conditional expectation given the information set available at time $t-1$, $\mathcal{I}_{t-1}$, and an orthogonal residual, $\varepsilon_{i,t}$:

$$r_{i,t} = \mathbb{E}[r_{i,t} \mid \mathcal{I}_{t-1}] + \varepsilon_{i,t}. \tag{1}$$

Our objective is to model the expectation of $r_{i,t}$ conditional on a set of K predictor variables $X_{i,t-1} \in \mathcal{I}_{t-1}$:

$$\mathbb{E}[r_{i,t} \mid \mathcal{I}_{t-1}] = g(X_{i,t-1}; \theta), \tag{2}$$

where $g(\cdot)$ is a flexible function of $X_{i,t-1}$ mapping firm characteristics to expected returns, parameterized through θ. Following Gu et al. (2020), we assume a common functional form across stocks and time.

2.2 Hyperparameters

Machine learning methods involve two types of parameters. *Model parameters* (θ_M) are estimated by minimizing a loss function using the training data. *Hyperparameters* (θ_H), in contrast, control the complexity of the model and be specified prior to model estimation. While estimating model parameters are straightforward conditional on hyperparameters, identifying hyperparameter values is complex.

In the context of return prediction, economic theory offers limited guidance on the choices

of hyperparameters. Thus, researchers often rely on values drawn from previous studies or perform grid searches over the pre-defined hyperparameter values. However, the configuration of this grid—its granularity and range—is seldom discussed in detail, even though it plays a critical role in shaping model performance. This sensitivity to hyperparameter settings is the key focus of our analysis.

One might ask why hyperparameters are not optimized jointly with model parameters? In practice, this turns out to be computationally infeasible. For example, if five hyperparameters are each evaluated at ten candidate values, the resulting grid contains 100,000 combinations. Researchers therefore constrain the search space using predetermined values appealing to "common practice", or use validation methods, an approach we revisit in Section 5.

To evaluate the sensitivity of return forecast performance to hyperparameter choices, we design a series of experiments in which one hyperparameter is fixed while the others are free to vary over their full grid. Specifically, we fix a single hyperparameter at a value, $\theta_{H1} = \theta_{H1,fix}$ and examine out-of-sample performance across the remaining hyperparameter space θ_{H1}^{C}. In each iteration, we split the data into two a training sample and a test sample. We use the training sample to estimate the model parameters based on a training loss function, $\mathcal{L}_{loss}$. The objective function, $\mathcal{L}(\theta)$, takes the general form:

$$\mathcal{L}(\theta) = \mathcal{L}_{loss}(\theta) + \mathcal{P}(\theta), \tag{3}$$

where $\mathcal{P}(\theta)$ is a penalty term. Following common practice (e.g., Gu et al., 2020), we use the mean squared error (MSE) as our baseline loss function:

$$\mathcal{L}_{l_2}(r_{i,t}, \theta) = \Big(r_{i,t} - g(X_{i,t-1}; \theta)\Big)^2. \tag{4}$$

The MSE loss function in (4) penalizes large errors heavily, making it sensitive to outliers. As a more robust training criterion we also consider the Huber loss function for the ensemble models and neural networks:

$$\mathcal{L}_\delta(r_{i,t}, \theta; \delta) = \begin{cases} \Big(r_{i,t} - g(X_{i,t-1}; \theta)\Big)^2, & \text{if } |r_{i,t} - g(X_{i,t-1}; \theta)| \leq \delta \\ 2\delta\Big(|r_{i,t} - g(X_{i,t-1}; \theta)| - \delta^2\Big), & \text{if } |r_{i,t} - g(X_{i,t-1}; \theta)| > \delta. \end{cases} \tag{5}$$

We set δ to the 99.9th percentile of the data, following Gu et al. (2020), which improves the stability of forecasts by limiting the influence of extreme values.

2.3 Linear Models

For linear machine learning algorithms we focus on penalized regression models, namely, the Lasso and Elastic Net. For these models, we have

$$g(X_{i,t-1}; \beta) = \beta' X_{i,t-1}, \tag{6}$$

such that the conditional expectation is approximated by a linear function of the predictor variables and parameter vector β. Penalized regression models minimize Equation (3), with the conventional least squares objective function and a penalty term. The Lasso approach includes an l_1 penalization and the Elastic Net includes both l_1 and l_2 penalization terms:

$$\mathcal{P}(\beta; \alpha, \lambda) = \begin{cases} \alpha \sum_{k=0}^{K} |\beta_k|, & \text{Lasso}, \\ \alpha\lambda \sum_{k=0}^{K} |\beta_k| + \frac{\alpha(1-\lambda)}{2} \sum_{k=0}^{K} \beta_k^2, & \text{Elastic Net}. \end{cases} \tag{7}$$

Here, α is the shrinkage hyperparameter on the coefficients and λ is the mixing hyperparameter determining how the l_1 and l_2 penalizations get weighted. For the Lasso approach $\lambda = 1$, so the loss function only depends on the penalty parameter α.

We do not tune the penalty parameters α and λ. Instead we use a grid of penalty parameters and generate one-month ahead forecasts recursively for each combination of hyperparameters. We refit the model every year, such that the parameters β are updated every twelve months. For Lasso we consider an evenly spaced grid of 20 values of the α hyperparameter between 0.0001 and 0.015. The lowest value (0.0001) corresponds to a very small penalty term while the upper value effectively excludes any predictor variables from being included, so higher values yield the same results. For Elastic Net we use the same grid for α while for λ we consider a grid between 0.2 and 0.8, with 0.05 increments, resulting in

13 settings. For λ there is a natural range between zero and one, so the main issue is how fine a mesh to use.[8]

2.4 Tree-based Models

Tree-based methods have been widely used in the ML literature. We focus on Random Forests and Gradient Boosted Trees, specifically Extreme Gradient Boosting, as regularization methods. Both offer non-parametric flexibility and are distinctive from traditional econometric methods.

Trees aim to identify groups of observations exhibiting similar behavior. A tree grows by sequential branching; data with similar behavior and characteristics are selected, forming the foundation for subsequent growth and branching of the tree. These branches, created in subsequent steps, group the remaining data from previous stages and organize them into bins or suitable intervals, approximating the functional form $g(\cdot)$ in Equation (2).[9]

At each node l of the tree, the algorithm evaluates potential splits by considering a subset of the predictors (features).[10] Let $F_l \subseteq \{1, 2, ..., K\}$ denote the set of features considered at node l, and $F = |F_l|$ the number of features. The splitting criterion is based on minimizing the impurity of the three, following an l_2 penalty quantified by the loss function $\mathcal{L}_{cart}(\theta_l, C_l)$:

$$\mathcal{L}_{cart}(\theta_l, C_l) = \frac{1}{|C_l|} \sum_{X_{i,t-1} \in C_l} \Big(r_{i,t} - \theta_l\Big)^2, \tag{8}$$

where $|C_l|$ is the number of samples at node l and the mean target value at node l is

$$\theta_l = \frac{1}{|C_l|} \sum_{X_{i,t-1} \in C_l} r_{i,t}. \tag{9}$$

For each feature $j \in F_l$ and potential split value s, the dataset C_l is divided into two

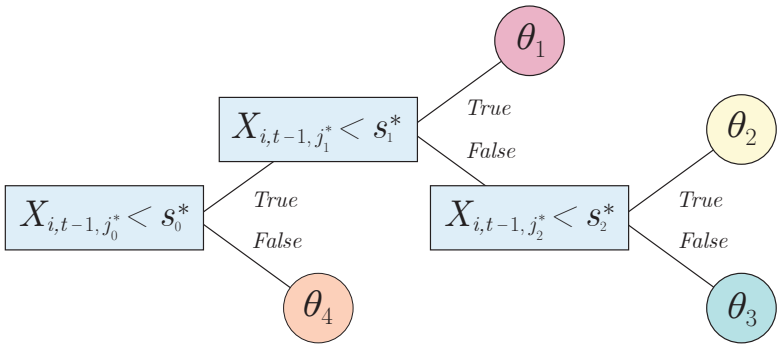
[8]When producing forecasts with the Lasso and the Elastic Net methods, we also allow a constant to be included in the feature space. However, at each point of re-estimation and for each penalty parameter, we allow the model the flexibility to determine whether or not to include the constant.

[9]Bagging and boosting algorithms both utilize classification and regression trees (CART) (Breiman et al., 1984) as the foundation for constructing individual trees. The defining characteristic of CART is its binary splitting process, where each node divides into exactly two child nodes.

[10]The growth of the tree starts with the root node l_0 which contains all samples in the training set, S_{l_0}.

Figure 1: Example of a Classification and Regression Tree

Example of a classification and regression tree with depth $D = 3$ and $L = 4$ leaves. Here, $X_{i,t-1,j}$ denotes the value of feature j for asset i at time $t-1$. Each node determines the optimal splitting feature j^* and value s^* via Equation (10). Leaf values θ_l represent the outcomes at each terminal node.



subsets $C_{l,1}(j,s)$ and $C_{l,2}(j,s)$. The optimal split (j^*, s^*) is then determined as

$$(j^*, s^*) = \arg\min_{j,s} \Big[\mathcal{L}_{cart}(\theta_{l,1}, C_{l,1}(j,s)) + \mathcal{L}_{cart}(\theta_{l,2}, C_{l,2}(j,s))\Big]. \tag{10}$$

This process is applied recursively until a stopping criterion is reached: when the tree expansion reaches the predefined maximum depth D, the number of samples at node l drops below S_l, or the resulting leaf nodes would have fewer samples than S_L.[11]

The final prediction is based on the average target values present in the samples of each leaf node in Equation (9). The prediction for a given feature set $X_{i,t-1}$ is computed using the combined contributions from all L leaf nodes of tree b:

$$\hat{r}_{i,t}^{(b)} = \sum_{l=1}^{L} \theta_l \, 1_{\{X_{i,t-1} \in C_l\}}, \tag{11}$$

where $1_{\{X_{i,t-1} \in C_l\}}$ is an indicator function which equals 1 when $X_{i,t-1}$ falls within the sample space of leaf node l, and is 0 otherwise.

A tree of depth D has the ability to capture $(D-1)$-way interactions but this flexibility of tree models comes at the cost of being highly prone to overfitting. Regularization techniques are used to overcome this problem. We consider Random Forests and Boosting. Both use

[11] As we re-estimate the model parameters, the splits and consequently θ_l depends on time t. For ease of notation, we omit the subscript t from θ_l.

ensemble methods to combine forecasts from a set of individual tree predictors.

2.4.1 Random Forest

The Random Forest (RF) method introduced by Breiman (2001) employs a bagging procedure over a collection of decision trees to minimize the variance in predictions. This process draws multiple bootstrap samples from the dataset and, for each sample $b = 1, ..., B$, constructs a regression tree based on the CART procedure.

The predictive power of the RF method comes from its ability to average forecasts from the individual trees, reducing the risk of overfitting and improving the overall predictive accuracy. The aggregated final prediction of the Random Forest is

$$g(X_{i,t-1}; \theta) = \frac{1}{B} \sum_{b=1}^{B} \hat{r}_{i,t}^{(b)}, \tag{12}$$

where $\hat{r}_{i,t}^{(b)}$ is the prediction from an individual tree in the forest.

Our choice of hyper parameter grid configuration focuses on evaluating the most influential hyperparameters that impact model performance (Probst et al., 2019):[12]

Number of trees in the forest, B. This hyperparameter influences the model's ability to generalize; more trees usually lead to better performance but also increase the computational complexity. We consider $B \in \{30, 100, 500\}$.

Maximum depth of a tree, D. A deeper tree can model more complex relationships by allowing more splits but also increases the risk of overfitting. D is a critical parameter for controlling the balance between bias and variance. We set $D \in \{1, 2, 4, 6, 8\}$.

Features in each split, F. The parameter F represents the number of features to consider at each split in the individual tree, dictating the size of the subset of features selected randomly from the total of K explanatory variables for evaluation at each decision node. We consider the grid $F \in \{1, 3, 10, 50, 70\}$.

[12] We also considered varying the hyperparameters for the minimum internal node size and the minimum node size. However, these do not have a significant impact on out-of-sample performance.

2.4.2 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost), an advanced implementation of Gradient Boosted Trees (Friedman, 2001), is a highly efficient and scalable algorithm for tree boosting. Unlike the traditional GBT that incrementally builds trees to minimize predictor bias through weak learners, XGBoost, developed by Chen and Guestrin (2016), applies gradient descent optimization within the context of boosting, enabling the recursive construction of trees. An advantage of XGBoost over methods like Random Forest is its approach to ensemble learning which involves optimizing Equation (8). This loss includes l_1 and l_2 regularization terms, crucial for controlling model complexity and preventing overfitting.[13]

The core of XGBoost lies in the sequential addition of trees with each tree designed to correct the residuals of all previous trees. These trees are commonly referred to as weak learners. In each boosting round b, a tree f_b is trained using the features $X_{i,t-1}$ with the objective to predict the residuals from the predictions made in round $b-1$. Here, $f_b(X_{i,t-1})$ denotes the output of the b-th tree. The learning rate, η, is crucial in this process—controlling the contribution of each new tree and ensuring a controlled learning process. The cumulative prediction, $\hat{r}_{i,t}^{(b)}$, is updated sequentially by adding the contribution of the b-th tree to the previous prediction;

$$\hat{r}_{i,t}^{(b)} = \hat{r}_{i,t}^{(b-1)} + \eta f_b(X_{i,t-1}). \tag{13}$$

The boosting process continues until the maximum number of rounds B is reached or other stopping criteria are met. Again, we consider a range of settings for the hyperparameters:

Number of trees in the forest, B. This hyperparameter influences the model's ability to generalize; more trees usually lead to better performance but also increases computational complexity. We consider $B \in \{1, 100, 500\}$.

Maximum depth of each tree in the forest, D. Deeper trees can model more complex relations by allowing more splits, but also increase the risk of overfitting. D is a critical

[13]Extreme Gradient Boosting constructs binary trees where each internal node is split into two child nodes. This methodology parallels the fundamental structure of CART, but XGBoost diverges from CART's greedy algorithm. Instead of making the best immediate split based on an MSE criterion, XGBoost uses a gradient boosting mechanism—allowing for an objective function of the general loss function form in Equation (3) with penalty term. This mechanism involves calculating the gradients and Hessian of the loss function, enabling the algorithm to efficiently determine the optimal splits by evaluating how effective each split is in reducing the overall loss.

parameter for controlling the balance between bias and variance. We set $D \in \{1, 2, 4, 6\}$.

Learning rate, η. The learning rate controls the step size of the prediction updating process. This parameter helps to ensure gradual improvement and prevent overfitting. We consider the grid $\eta \in \{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4\}$, which includes the default value of 0.3 and covers the range often found to be optimal in empirical applications, typically between 0.01 and 0.2 (see, e.g., Chen and Guestrin (2016) and the corresponding XGBoost documentation).

We use histogram-based boosting as the booster method, which is a fast implementation that approximates the greedy algorithm. The l_1 and l_2 regularization on the weights are set to $\alpha = 0$ and $\lambda = 1$, respectively.[14]

Further, *early stopping* is a well-known technique employed in machine learning to prevent overfitting and enhance model generalization. The model is trained on a training set, and the performance is evaluated on a validation set. The training process is halted when the model's performance on the validation dataset ceases to improve or starts to degrade, ensuring that the model does not become overly specialized to the training data. We set the patience parameter to 5, indicating the number of consecutive iterations during which no improvement is observed in the model's performance—see Algorithm 1. When early stopping is used, we do not retrain the model on both the training and validation sets and instead use the model to directly make predictions on the test set. This approach helps ensure that the model's evaluation on the test set remains consistent with its ability to generalize to unseen data.

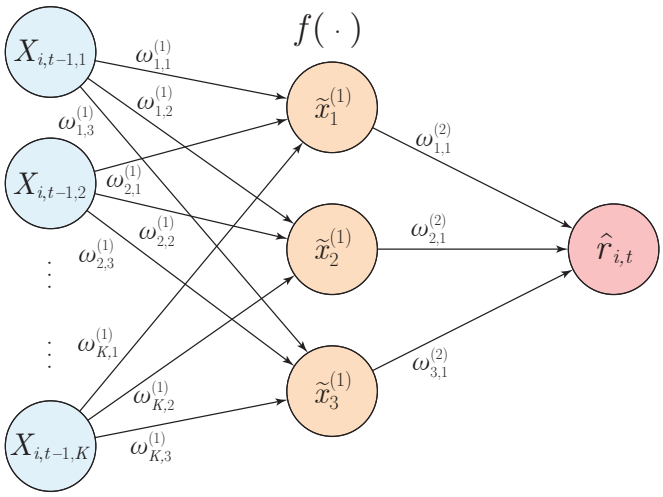
2.5 Neural Networks

Neural networks are anchored in their ability to approximate a broad range of continuous functions (Kolmogorov, 1957; Cybenko, 1989; Hornik et al., 1989). Specifically, a Feed-Forward Neural Network (FNN) with a single hidden layer and a finite number of neurons can approximate any continuous function on compact subsets of real numbers, given appropriate parameters and activation functions. In FNNs, data travels exclusively forward starting from

[14]Note that in the context of XGBoost, the parameters α and λ have definitions that differ slightly from their usage in the Elastic Net. Here, they directly correspond to the l_1 and l_2 penalization, respectively. Additionally, we investigate a range of values for α and λ, as well as the proportions of observations and features randomly sampled for each tree. However, these values do not affect the model's performance.

Figure 2: Example of a Feed Forward Neural Network

Example of a Feed Forward Neural Network with $H = 1$ hidden layer. There are K blue input nodes corresponding to each feature, $X_{i,t-1,k}$. The hidden layer contains 3 neurons, marked in orange, and the output layer is the pink node. The function $f(\cdot)$ in the hidden neurons denotes the activation function.



the input layer, passing through (multiple) hidden layers, and aggregating in the output layer. Each neuron in the hidden layers represents a distinct feature of the data. The initial input layer receives the predictor input, which then gets processed and transformed non-linearly through H hidden layers. The output layer aggregates the transformed data, yielding a prediction. Thus, $g(\cdot)$ in Equation (2) is approximated by the hidden layers. Figure 2 shows a visual representation of a FFN with one hidden layer containing three neurons.

We construct the FNNs in the following way. Let $\widetilde{x}_k^{(\ell)}$ denote the output coming from neuron k in layer ℓ, and let $\widetilde{x}^{(\ell)} = \left(\widetilde{x}_1^{(\ell)}, ..., \widetilde{x}_{K^\ell}^{(\ell)}\right)'$, with K^ℓ being the number of neurons in layer ℓ. The input layer takes data $\widetilde{x}^{(0)} = X'_{i,t-1}$ and forwards it to the fully connected first hidden layer. We recursively construct the output of each neuron k in layer $\ell > 0$ as

$$\widetilde{x}_k^{(\ell)} = f\left(\widetilde{x}^{(\ell-1)\prime}\omega_k^{(\ell)} + \omega_0^{(\ell)}\right), \tag{14}$$

where $\omega_k^{(\ell)}$ are the weights of neuron k in layer ℓ. The bias term $\omega_0^{(\ell)}$ in layer ℓ allows the activation function to shift, enabling the model to effectively fit the data regardless of input

values. Further, $f(\cdot)$ is the activation function. Finally, the prediction $\hat{r}_{i,t}$ is computed as

$$g(X_{i,t-1};\theta) = \widetilde{x}^{(H)\prime}\omega^{(H+1)} + \omega_0^{(H)}, \tag{15}$$

where $\widetilde{x}^{(H)}$ denotes the output from the last hidden layer, $\omega^{(H+1)}$ the weights linking these to the final layer and $\omega_0^{(H)}$ the final layer bias term, summarizing the contributions of each neuron in the last hidden layer for the prediction.

To determine the optimal weights of the neural nets $\omega_k^{(\ell)}$, we consider two loss functions, namely mean squared error of Equation (4) and Huber loss of Equation (5). Determining the optimal weights of a neural network is often computationally inefficient using standard stochastic gradient descent. We use the Adaptive Moment Estimation (Adam) optimizer from Kingma and Ba (2014), which enhances efficiency by adjusting the learning rate for each parameter during training. Let ω_d represent the weights and biases across all layers at step d. The weight update mechanism at step d operates as follows;

$$\omega_d = \omega_{d-1} - \frac{\eta \cdot \hat{m}_d}{\sqrt{\hat{v}_d} + \epsilon}, \tag{16}$$

where η is the Adam learning rate, $\hat{m}_d$ and $\hat{v}_d$ are bias-corrected estimates of the gradient's first and second moments of the loss function, and $\epsilon = 10^{-8}$ prevents division by zero. The Adam optimizer is shown in Algorithm 2.

We follow Gu et al. (2020) and consider five types of FNNs, labeled FNN1 through FNN5, corresponding to $H = 1$ to $H = 5$ hidden layers, respectively. The number of neurons decreases by half for each subsequent layer and follow a geometric pyramid that starts from 32. FNN1 thus has a single layer with 32 neurons and FNN5 features five layers with 32, 16, 8, 4, and 2 neurons, respectively. As in Figure 2, all layers are fully connected such that every neuron receives its input from the preceding neurons.

Activation function $f(\cdot)$. We focus on the linear activation function and the rectified linear unit (ReLU) activation function; ReLU $= \max(0, x)$. The ReLU function allows for faster derivative evaluation and favors sparsity. However, ReLU can result in zero outputs, stopping learning in some neurons. To mitigate this, we also use Leaky ReLU $= \max(0.01x, x)$,

which maintains a nonzero gradient for inactive units, ensuring continuous learning.[15]

Adam learning rate η. The learning rate of the optimizer in Equation (16), is critical for performance, as demonstrated by Bergstra and Bengio (2012) on seven datasets. We consider the set $\eta \in \{0.0001, 0.001, 0.01, 0.1\}$, which includes the default value of 0.01 as suggested by Kingma and Ba (2014) and contains and extends those used in prior financial literature (Table 1).

Shrinkage penalty α. As neural nets are heavily parameterized, some form of regularization needs to be introduced. We therefore introduce an l_1 penalty term on the weights in the loss function, similar to the Lasso approach. In practice, $\alpha = 0.01$ is often used. We consider $\alpha \in \{0.0001, 0.001, 0.01, 0.1\}$.

As in XGBoosting, we also perform *early stopping* in the feed-forward neural networks. Amari et al. (1995) show that early stopping in neural networks always decreases the generalization error, leading to better performance on unseen data. We use a validation set for the stopping criteria. If the objective function does not improve after 5 iterations, we stop the training. For training, we use 100 epochs.[16]

2.6 Summary of Hyperparameter Ranges

For each model type, Table 2 summarizes which hyperparameters we vary along with the range of values taken. The total number of hyperparameters under consideration is obtained, for each model type, as the product of each set of hyperparameter listed in the table and ranges from 20 for the Lasso approach to 260 for Elastic Net models.

3 Data

Our data consists of monthly values of firm-level stock returns and characteristics obtained from CRSP and Dacheng Xiu's website, respectively. We measure returns in excess of the

[15] The Sigmoid and Tanh function are also common choices as activation functions. However, these functions are susceptible to saturation with low signal-to-noise input, leading to a vanishing gradient (Glorot and Bengio, 2010). This limits the model's ability to learn from weak signals.

[16] Prechelt (2002) shows that increasing the patience results in small improvements at the cost of increased computation time. Further, increasing the number of epochs has minimal effect on the performance.

Table 2: Ranges of hyperparameters for the ML models

Model Type	Model	Hyperparameter		Grid size	Grid range
Linear	Lasso	Shrinkage strength	α	20	$\{0.0001, 0.009, ..., 0.015\}$
Linear	Elastic Net	Shrinkage strength	α	260	$\{0.0001, 0.009, ..., 0.015\}$
		(l_1, l_2) penalty mix	λ		$\{0.2, 0.25, ..., 0.8\}$
Tree-based	Random Forest	Number of trees	B	75	$\{30, 100, 500\}$
	(RF)	Maximum depth of tree	D		$\{1, 2, 4, 6, 8\}$
		Number features in split	F		$\{1, 3, 10, 50, 70\}$
Tree-based	Extreme Gradient	Number of trees	B	84	$\{1, 100, 500\}$
	Boosting (XGB)	Maximum depth of tree	D		$\{1, 2, 4, 6\}$
		Learning rate	η		$\{0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4\}$
Neural Nets	Feed Forward	Activation functions	$f(\cdot)$	96	{`Leaky`, `linear`, `ReLU`}
	NN 1–5	Loss function	$\mathcal{L}(\cdot)$		{`MSE`, `Huber`}
		Adam learning rate	η		$\{0.0001, 0.001, 0.01, 0.1\}$
		l_1 shrinkage penalty	α		$\{0.0001, 0.001, 0.01, 0.1\}$

risk-free rate proxied by the 1-month Treasury Bill rate obtained from Kenneth French's website. As in Gu et al. (2020), we assume that monthly, quarterly, and annual data are accessible to investors with one-month, four-month, and six-month lags. Details on the frequency and description of the characteristics are reported in Table A.1.

The dataset contains many missing entries. For every month we only consider firms from which the monthly return is available since our goal is to predict excess returns. Although the original data goes further back in time, to balance the cross-sectional and time-series dimension, we choose a dataset that ranges from January 1977 to December 2021, contains 92 firm- or stock-level characteristics and 48 industry-specific dummies, resulting in 140 features.[17] Before 1985, there are between 300 and 800 firms in our dataset each month. After 1985, the number of firms averages 1500 per month. In total there are 6,840 unique firm numbers.

3.1 Missing Characteristics and Data Transformations

Missing data points are typically not random draws and so need to be carefully handled. Data missing earlier in our sample arises primarily due to coverage issues while missing data later in the sample occur mostly because of the Global Financial Crisis (GFC) and the COVID pandemic. To address the issue of missing characteristics in our dataset, we follow

[17]The original panel of characteristics goes back to January 1957. However, data prior to 1977 has many missing entries.

Bryzgalova et al. (2024) and consider data from January 1977 onward since the frequency of missing characteristics is significantly lower after 1976. Furthermore, we consider 92 characteristics instead of 94, since two of these (real estate holdings and secured debt) are only available from 1985 onward, and even then, only sparsely available across time.

Next, we employ the local backward (B-XS) model of Bryzgalova et al. (2024) which fills data by using contemporaneous cross-sectional information and recent historical characteristics.[18] The B-XS model is the best-performing model that does not use forward-looking information for imputing characteristics in terms of out-of-sample mean squared error. For cross-sectional fitting we use the settings of Bryzgalova et al. (2024). That is, we capture the cross-sectional dependence using six latent factors, we only consider stocks that have at least ten characteristics observed at each point in time, and we use the covariance structure between the characteristics of January 1985 to avoid look-ahead bias. After converting the characteristics to rank percentiles, we impute the missing data and then transform the imputed rank quantiles back to their original values using the empirical distribution.[19]

The characteristics are measured on different scales and so are not directly comparable. To handle this issue, we cross-sectionally rank the explanatory variables for each period and scale them to the $[-1, 1]$ range, as proposed in Gu et al. (2020). This method is more robust to outliers than conventional standardization method because it imposes boundaries at -1 and 1. However, it also has some drawbacks. Cross-sectionally ranking the variables between -1 and 1 erases information about the level of the variables. We further construct dummies based on the first two digits of the Standard Industrial Classification (SIC) codes.[20] To obtain as granular a grouping as possible, but with a sufficient number of firms, we group some of the first two digits of the SIC numbers which results in 48 dummies.[21]

[18] We are grateful to Markus Pelger and Sven Lerner for sharing their code with us.

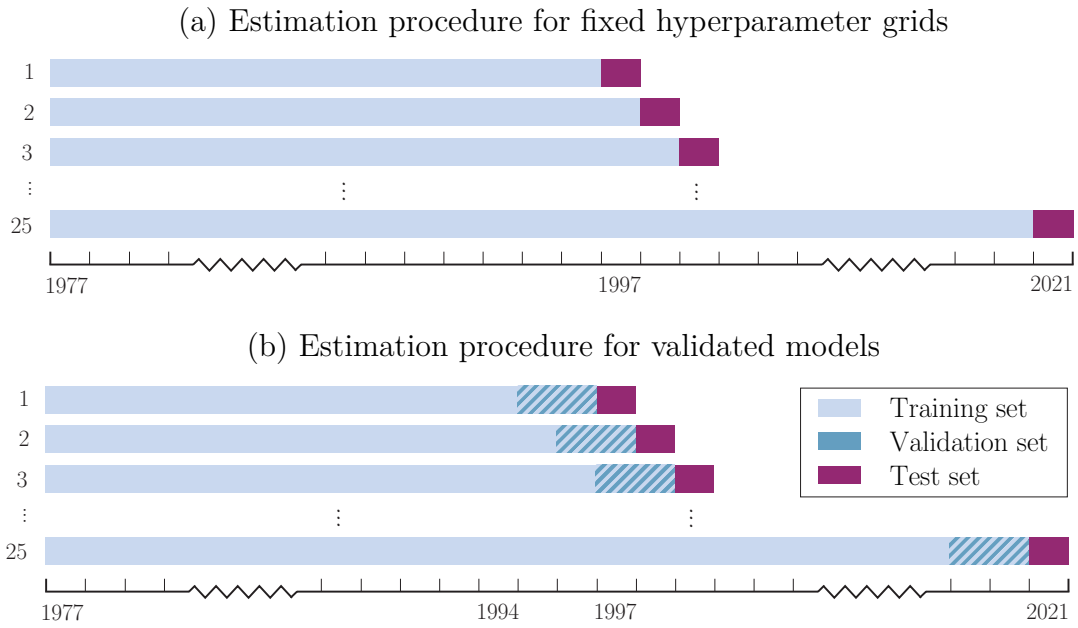
[19] Note that we handle missing data very differently from Gu et al. (2020) who fill firm-level missing characteristics with zeros.

[20] Appendix Table A.2 shows the grouping we use.

[21] We exclude macroeconomic predictors because the scaling process eliminates the information these variables contain. Because our most flexible ML models can capture complex non-linear relations and interaction terms we do not take the Kronecker product between characteristics and macro variables.

Figure 3: Training, Validation, and Test Samples

Panel (a) shows the training (estimation) window for the procedure that uses fixed hyperparameters. The model undergoes training using the training dataset, shown in light blue, and performs out-of-sample predictions on the test dataset, shown in purple. Panel (b) shows the estimation process incorporating a 2-year validation period. If validation or early stopping mechanisms are applied, the model is trained with the training set (light blue) and validated on the validation set (dark blue). If early stopping is not employed, the model parameters are re-estimated using both the training and validation sets before making out-of-sample predictions on the test set (purple).



3.2 Training and Test Set

To examine the sensitivity of our return predictability results with respect to the values of the hyperparameters, we split our data into training and test sets. The training set is used to estimate the model parameters and our initial training uses the first 20 years from January 1977 to December 1996. The test sample evaluates the performance of the model on unseen data and is the 25-year out-of-sample period from January 1997 to December 2021 (300 months). Using an expanding estimation window, we update the model parameters every 12 months as visualized in the upper panel in Figure 3. The validation procedure (shown in the lower panel of Figure 3) is discussed in Section 5.

3.3 Evaluation of Forecasts

We evaluate predictive performance by means of the widely used out-of-sample R^2 measure of Campbell and Thompson (2008) generalized to account for multiple stocks:

$$R^2_{OoS} = 1 - \frac{(T-T_0)^{-1}\sum_{i=1}^{N_t} N_t^{-1}\sum_{t=T_0+1}^{T}\left(r_{i,t}-\hat{r}_{i,t}^{(m)}\right)^2}{(T-T_0)^{-1}\sum_{i=1}^{N_t} N_t^{-1}\sum_{t=T_0+1}^{T}\left(r_{i,t}-\hat{r}_{i,t}^{(bm)}\right)^2}, \tag{17}$$

where T_0 is the start of the test sample which ends at time T, N_t is the number of firms in existence at time t, $\hat{r}_{i,t}^{(m)}$ denotes the predicted excess returns from model m, $\hat{r}_{i,t}^{(bm)}$ is the benchmark forecast, and $r_{i,t}$ is the realized excess return of stock i at time t. R^2_{OoS} is a relative measure tracking the predictive accuracy of model m relative to the benchmark forecasts with positive values indicating outperformance. As in Gu et al. (2020), we use zero prediction as the benchmark model, i.e., $\hat{r}_{i,t}^{(bm)} = 0$ for all i and t.[22]

We use the Diebold-Mariano (DM) test (Diebold and Mariano, 1995) to test for significance in the differences between the out-of-sample predictive accuracy of our models and the zero prediction. Specifically, we use a cross-sectional adaptation of the DM test to test forecast accuracy. The prediction error differential for time t is defined as

$$d_t = \frac{1}{N_t}\sum_{i=1}^{N_t}\left(\left(\hat{e}_{i,t}^{(bm)}\right)^2 - \left(\hat{e}_{i,t}^{(m)}\right)^2\right). \tag{18}$$

where $\hat{e}_{i,t}^{(bm)} = r_{i,t}$ and $\hat{e}_{i,t}^{(m)} = r_{i,t} - \hat{r}_{i,t}^{(m)}$. The DM statistic for model m is then $DM_m = \overline{d}_m / \hat{\sigma}_{\overline{d}_m}$, where $\overline{d}_m$ is the time series average of d_t and $\hat{\sigma}_{\overline{d}_m}$ is the Newey-West standard error.

To track the time-series evolution in the predictive accuracy of model m versus the benchmark, we also plot the cumulative sum of squared error differences $CSSED_\tau = \sum_{t=t_0}^{\tau} d_t$ over the test set.

[22] Alternatively, one could use the prevailing mean, $\overline{r}_{i,t} = \frac{1}{t-1}\sum_{\tau=0}^{t-1} r_{i,\tau}$, which is the average excess return of firm i until $t-1$, as a benchmark. However, the prevailing mean is noisy and generates a 4.5% higher MSE compared to the zero prediction for individual stocks.

4 Empirical Results

In this section, we report the forecasting performance of the models introduced in Section 3. We summarize the sensitivity of our R^2_{OoS} measure of predictive accuracy with respect to hyperparameter values through boxplots constructed by fixing one hyperparameter at a given value while allowing the remaining hyperparameters to vary across the grid values listed in Table 2. Each boxplot, complete with whiskers, displays the range of R^2_{OoS} values achievable as we vary the other hyperparameters within a predetermined grid. Our boxplots follow standard practice: the central box encompasses the interquartile range (IQR) between the 25th percentile, $q_{0.25}$, and the 75th percentile, $q_{0.75}$, so $IQR = q_{0.75} - q_{0.25}$. The line inside the box corresponds to the median, $q_{0.50}$. Whiskers visualize the range of R^2_{OoS} values outside the middle 50%, using the $1.5 \times IQR$ rule. Hence, the lower whisker extends from the first quartile ($q_{0.25}$) to the lowest data point not below $q_{0.25} - 1.5 \times IQR$ while the upper whisker goes from the third quartile ($q_{0.75}$) to the highest data point not above $q_{0.75} + 1.5 \times IQR$. Values outside the boundaries of the whiskers are plotted as individual points and are considered outliers.[23]

4.1 Linear Models

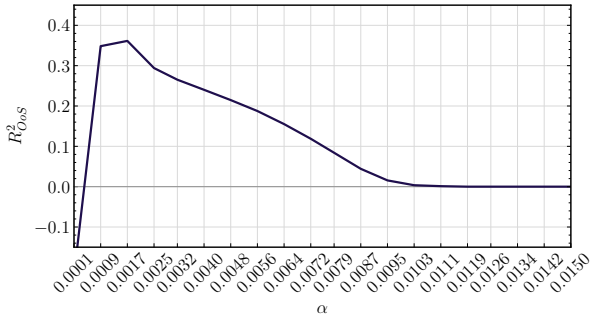
Figure 4 panel (a) shows the sensitivity of the out-of-sample performance of Lasso across values of the single hyperparameter, α, which controls the amount of shrinkage with larger values of α corresponding to a higher level of regularization. The R^2_{OoS} ranges from -0.2 for penalty terms approaching zero to 0.38 for small α values. The R^2_{OoS} curve is flat at zero for $\alpha > 0.01$ because the Lasso approach does not select any predictor variables. Supporting this interpretation, Panel (b) of Figure 4 plots the percentage of non-zero coefficients over the estimation window. When $\alpha > 0.01$, the estimated coefficients for all models are reduced to zero across every window.

[23]For example, for a random forest model we fix hyperparameter B at 30, and let the hyperparameters D and F vary over the grid, which could result in the following range of R^2_{OoS} values: $\{-0.05, 0.18, 0.22, 0.23, 0.28, 0.3, 0.35, 0.39, 0.41\}$. The corresponding central box ranges from 0.22 to 0.35, as the corresponding percentiles are $q_{0.25} = 0.22$ and $q_{0.75} = 0.35$, and $IQR = 0.13$. The median is 0.28. The lower whisker is determined by the smallest value still above $q_{0.25} - 1.5 \times IQR = 0.025$, which is 0.18. The upper whisker is 0.41, which is the highest value still below $q_{0.75} + 1.5 \times IQR = 0.545$. The value -0.05 is shown by a dot.

Figure 4: Sensitivity of R^2_{OoS} to Lasso hyperparameter (α)

Panel (a) shows the sensitivity of R^2_{OoS}, expressed as a percentage, with respect to different choices of α. Panel (b) shows the percentage of variables $X_{i,t}$ included by Lasso for different values of α.

(a) Sensitivity of R^2_{OoS} to l_1 shrinkage α



(b) Percentage of selected variables

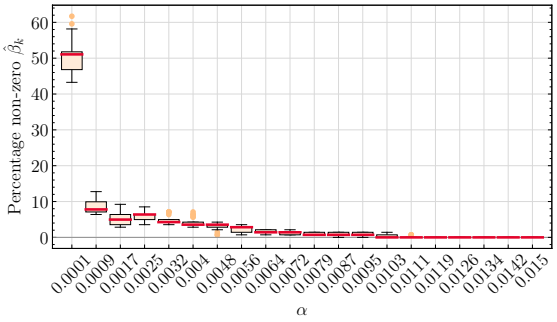
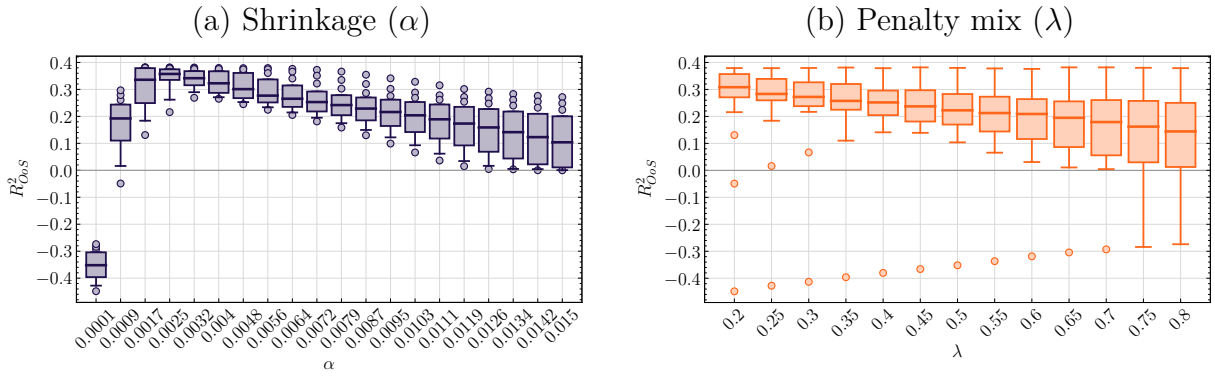


Figure 5 shows the sensitivity of R^2_{OoS} for the Elastic Net across the range of values for the two hyperparameters, α and λ. Panel (a) shows the sensitivity of R^2_{OoS} when keeping the amount of shrinkage (α) fixed at a given value while varying the (l_1, l_2) penalty mix λ. For $\alpha = 0.0001$, the range of R^2_{OoS} values is between -0.5 and -0.25 consistent with underperformance against the zero benchmark. Performance improves when α is fixed around 0.001 and gets a further boost once we increase α to 0.003 or 0.004 which raises the average R^2_{OoS} value and narrows its range so that even the worst model across the grid of λ values performs better than the median performer for $\alpha = 0.001$. Increasing α beyond these values pulls the R^2_{OoS} towards zero and expands the range of R^2_{OoS} values, thus increasing the risk of selecting a model with poor forecasting performance.

Panel (b) of Figure 5 shows the range of R^2_{OoS} values for a given value of the (l_1, l_2) penalty mix parameter λ while varying α. The performance of both the median and best models is less sensitive to the choice of λ than to the choice of α. Outliers marked by dots occur for the lowest value $\alpha = 0.0001$ which results in too many predictors being selected and noisy forecasts. Higher values of λ tend to increase the minimum R^2_{OoS} value.

Figure 5: Sensitivity of R^2_{OoS} to hyperparameters for Elastic Net

The figure displays the sensitivity of the out-of-sample R^2_{OoS} in percentage for the panel of individual stocks on the test sample from January 1997 to December 2021 for the Elastic Net. Panel (a) varies the level of shrinkage (α) while Panel (b) varies the penalty mix parameter (λ).



4.2 Tree-based Models

Figure 6 panel (a) presents the range of R^2_{OoS} values for the random forest models whose hyperparameters are B, the number of trees averaged over by the random forest; D, the maximum depth of each tree; and F, the number of features considered for each split. Out-of-sample performance for the median, top-quartile and best model is robust with regards to the choice of the number of trees (B). Conversely, the worst and bottom-quartile models perform notably poorer when $B = 30$ than when $B = 500$. The maximum depth of the trees (D) is more important. For $D = 1$, the range of R^2_{OoS} values is narrow and centered around 0.30. Increasing D shifts the median value of R^2_{OoS} down and increases the range of values so that, when $D = 6$, the median model generates a negative R^2_{OoS}. A similar pattern is observed from increasing the number of features in the split, F. When $F = 1$, the range of R^2_{OoS} values is narrow and centered around 0.30. Increasing F beyond this value reduces the average R^2_{OoS} and expands its range.[24] Limiting the number of features to just one per tree increases diversity among the trees in the ensemble, in contrast to using many features which can lead to trees splitting on similar information. This increased diversity strengthens model generalization, resulting in a consistently narrow range of performance metrics as the Random Forest averages the outputs of these diverse trees.

[24]The minimal internal and leaf node size barely affect the out-of-sample performance.

Figure 6: Sensitivity of R^2_{OoS} to hyperparameters for tree-based models

This figure shows the sensitivity of the out-of-sample R^2_{OoS} in percentage for the panel of individual stocks from January 1997 to December 2021 for Random Forest and Extreme Gradient Boosted Trees. For Random Forests, Panel (a) displays the sensitivity to the number of trees B, tree depth D, and the number of features F. For Extreme Gradient Boosted Trees Panel (b) shows the sensitivity to the number of trees B, tree depth D, and the learning rate η. The Boosted trees are trained with early stopping with a patience of 5, where the early stopping criteria are based on the lowest MSE within a 2-year validation set.

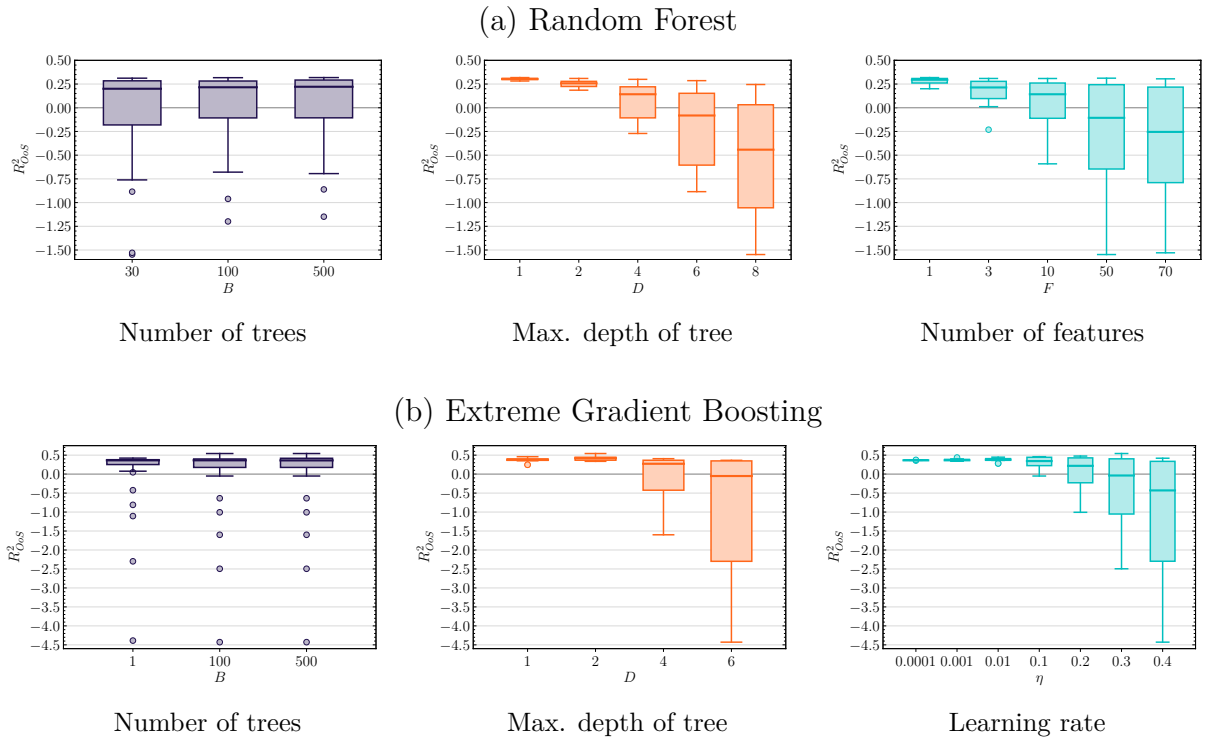


Figure 6 panel (b) shows the R^2_{OoS} for the Extreme Gradient Boosted Regression Trees.[25] The number of trees (B) has little effect on predictive performance. In contrast, the maximum depth of the trees and the learning rate are both important. Shallow trees (low D) perform best and settings $D = 4, 6$ result not only in a decline in the median model's performance but also widens the range of R^2_{OoS} values with the bottom-quartile model having a negative R^2_{OoS} value. The learning rate also has a large impact on R^2_{OoS} values with $\eta = 0.3, 0.4$ leading to poor performance for the bottom-quartile models which now generate a R^2_{OoS} value close to -1 and -2, respectively.

The interplay between the learning rate and tree depth in XGBoost is key to understanding its out-of-sample forecasting performance. A combination of low depth and a high

[25] Note that the specified number of trees does not reflect the actual number of constructed trees or boosting rounds since we use early stopping.

learning rate generally strikes a good balance between model complexity and learning efficiency. Low depth together with a low learning rate tends to produce overly conservative models and is indicative of underfitting. High depth with a low learning rate results in conservative performance, while a high depth alongside a high learning rate typically leads to poor model performance due to overfitting and instability in the learning process.

4.3 Neural Networks

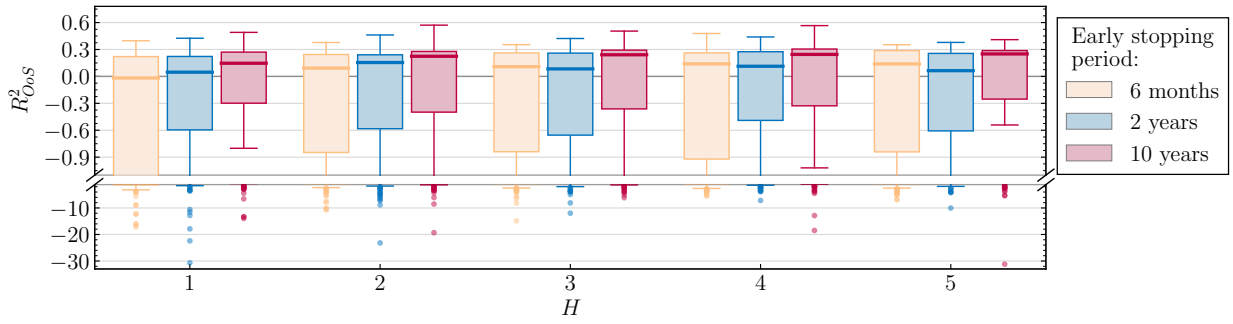
We finally turn to the NN models. Panel (a) in Figure 7 shows out-of-sample forecasting performance across NN models that fix the number of hidden layers H between 1 and 5 and vary all other hyperparameters, resulting in a total of 96 settings per NN specification. The box plots feature a broken y-axis to accommodate the wide variation in performance. The lower segment of the axis is deliberately capped at -30. For each hidden layer count (H), the figure features three box plots: the first utilizes a 6-month set for early stopping, the second a 2-year set, and the third a 10-year set for this purpose. For all architectures, both the median and the worst-performing models exhibit better out-of-sample performance on the 2-year set compared to the 6-month and 10-year period. The improved out-of-sample performance achieved with a 2-year early stopping period suggests that there is a trade-off between incorporating more recent data into the training process and maintaining a sufficiently large early stopping set. Moreover, the use of a larger (smaller) early stopping sample has the potential to result in underfitting (overfitting) during training, consequently leading to a decline in out-of-sample performance.

Figure 7 panel (b) focuses on the performance of a specific network architecture, NN2, over a 2-year validation period. To conserve space, we only report the NN2 architecture for the 2-year validation set as the results are very similar. The ReLU activation function performs best in terms of median performance and downside risk. Further, the out-of-sample performance is not much affected by the training loss function. For both validation periods, choosing the slowest learning rate $\eta = 0.0001$ or the second slowest rate $\eta = 0.001$ yields the best results overall. For both the 6-month and 10-year validation period, performance can be preserved by choosing a very slow learning rate. The performance differential between

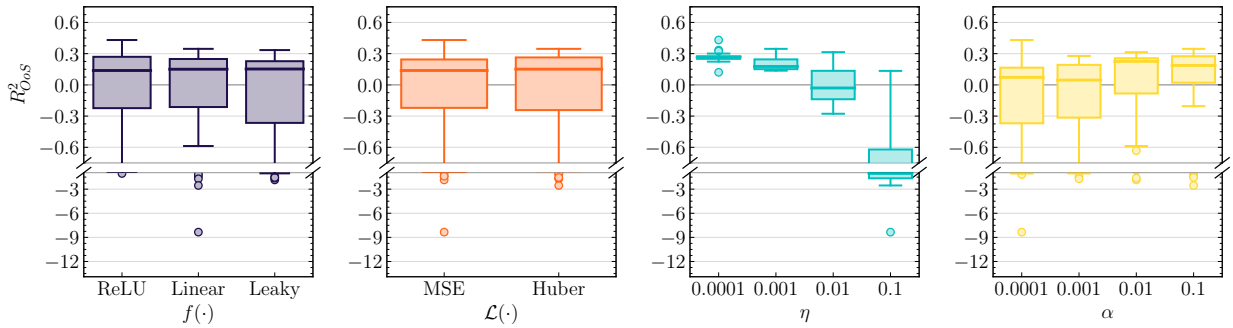
Figure 7: Sensitivity of R^2_{OoS} to architecture and hyperparameters for the Feed-Forward Neural Networks

This figure shows the out-of-sample R^2_{OoS} sensitivity, expressed in percentage, for the range of individual stocks from January 1997 to December 2021. Panel (a) shows the sensitivity across architectures and different set sizes used to define the early stopping criteria. The y-axis is capped at -30%. Panel (b) shows the sensitivity across the hyperparameters of Neural Networks with $H = 2$ hidden layers, utilizing a 2-year period to implement early stopping. The y-axis is capped at -12%. All models apply early stopping with a patience of 5 based on achieving the lowest MSE during this period.

(a) NN1 to NN5 with varying early stopping period



(b) NN2 with a 2-year early stopping period



the validation periods can be linked to the l_1 shrinkage penalty α. For both the 6-month and 10-year validation period, the smaller value of $\alpha = 0.0001$ results in underperformance, likely due to overfitting. All in all, the best approach here is to have a smaller validation set of for example 2-years, a relatively deep neural net architecture and apply a slow learning rate, with some regularization.[26]

[26] In deeper neural networks with 4 or more layers, predictions become constant due to large biases $\omega_0^{(\ell)}$ dominating the effects of inputs multiplied by weights in Equation (14), pushing activation functions into saturation. This saturation results in outputs that are insensitive to input variations, thereby hindering the network's ability to learn effectively from data.

4.4 Summary of Results

We summarize the key insights from our analysis as follows. First, our results show that it is possible to fix a subset of the hyperparameters such that good out-of-sample forecasting performance is achieved regardless of how the remaining hyperparameters are chosen. For the RF method this occurs when the maximum depth D equals 1 or 2 or when only a single feature is included ($F = 1$). For XGBoost, this is achieved when the maximum depth $D = 1, 2$ or when the learning rate is very low. For NNs, similarly a very slow learning rate is likely to lead to accurate forecasts regardless of how the other hyperparameters are set.

Second, the opposite also holds: the hyperparameters can be set so that poor forecasting performance is, if not guaranteed, highly likely even when the regular parameters are chosen optimally. For penalized linear models such as the Lasso or Elastic Net, this happens when the shrinkage parameter (α) is very low. For the RF models this occurs when the maximum depth or the number of features are high (e.g, $D = 8$ or $F = 70$) and for XGBoost when $D = 6$ or the learning rate, η, is high. Similarly, a high learning rate tends to "noise up" forecasts generated by NNs, leading to poor out-of-sample forecasting performance regardless of how other hyperparameters are chosen.

Third, due to their extra flexibility, varying the grid of hyperparameters of the nonlinear ML models tends to expand the range of R^2_{OoS} values far more than that generated by varying the shrinkage and penalty hyperparameters of the linear models. This widening in the range of out-of-sample performance means that extra caution has to be exercised when evaluating the predictive accuracy of ML methods to avoid spurious outcomes even after accounting for the tendency of out-of-sample forecasting methods towards being conservative.

Fourth, The widening of the range of out-of-sample forecasting performance of ML models relative to conventional linear models is highly asymmetric and left-skewed. In particular, we see a clear upper limit on the maximum degree of outperformance achievable by these models while, conversely, ML models with hyperparameters chosen so they are sensitive to noise are prone to generate extremely poor out-of-sample forecasts. This pronounced "downside" risk calls for caution by investors and portfolio managers using these models to generate forecasts of asset returns.

5 Validation Methods

So far we demonstrated the sensitivity of the predictive accuracy of return forecasts to the choice of hyper parameters by fixing the latter across a range of values as shown in Table 2. In this section, we present model performance using time series validation and the economic value of return forecasts.

Because there is little theoretical guidance on selecting hyperparameters, it is common practice in finance (e.g., Gu et al., 2020; Bianchi et al., 2021; Bali et al., 2023) to undertake time-series validation on a pseudo out-of-sample period. Validated models use data to determine the hyperparameters based on performance in the validation sample, rather than setting them a priori. The validation sample is formed by partitioning the original training sample into a training and validation sample, and is used to tune hyperparameters. The objective of the validation procedure is to determine the optimal set of hyperparameters by identifying the configuration that yields the best fit in the validation set.

In this section, we set out to examine whether validation methods can be used to select hyperparameters in a way that avoids poor out-of-sample forecasting performance.

5.1 Validation Analysis and Results

Validation methods do not offer a fully automated solution to the hyperparameter selection problem. Researchers must choose whether to use time-series validation or cross-validation, the length of the validation window, how to split a given sample into training, validation and test samples (sample split), and establish the initial hyperparameter grid for conducting the validation procedure. Lastly, researchers need to decide whether early stopping is suitable or not.[27] All of these choices affect out-of-sample forecasting performance.

Following common practice in finance, we use time-series validation in our analysis as illustrated in the bottom panel of Figure 3. To tune hyperparameters, we consider validation sets with lengths of 6 months, 2 years, and 10 years, covering the common choices in literature

[27] If early stopping is implemented, the model is trained solely on the training set with the optimal hyperparameters without re-estimating the model parameters on the combined training and validation sets. Conversely, if early stopping is not used, the model is re-estimated using both the training and validation sets after determining the optimal hyperparameters.

Table 3: Performance statistics for the validated models

We report the monthly R^2_{OoS} in percentage, and the corresponding Diebold-Mariano (DM) statistic for the validated models with different validation periods, where the MSE is used as the evaluation metric. For XGBoost and neural networks, e.s. refers to early stopping which is based on the same period as the validation procedure.The DM statistic is asymptotically normally distributed.

	Lasso		Elastic Net		Random Forest		XGBoost (e.s.)		NN2 (e.s.)		NN5 (e.s.)	
	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}
6 months	0.401	0.069	0.323	0.059	-0.475	-0.242	-1.398	-0.795	-3.294	-2.753	-3.755	-3.204
2 years	0.624	0.153	1.056	0.254	0.160	0.013	0.648	0.221	-1.310	-0.451	-1.836	-0.669
10 years	0.828	0.170	0.395	0.081	0.526	0.143	0.636	0.197	0.785	0.195	1.170	0.282

(Gu et al., 2020; Bali et al., 2023). The shorter validation set ensures that the model captures the latest information while the 10-year set captures a broader range of economic conditions which could improve the model's ability to generalize across different scenarios. We use the hyperparameter grids shown in Table 2 for the the validation exercise.

Table 3 reports R^2_{OoS} values for different validation windows. The length of the validation window is crucial to the models' performance. Specifically, tree-based models and neural networks exhibit subpar performance with a validation set of 6 months. For longer validation windows such as 10 years, performance is generally similar to or worse than for a 2-year period although NNs show more robust performance on the 10-year validation set.

Table 3 also reports the associated DM-statistics. None of the validated models significantly outperform the zero benchmark throughout the out-of-sample test sample. Conversely, the NN2 and NN5 models based on the shortest validation window significantly underperform this benchmark.

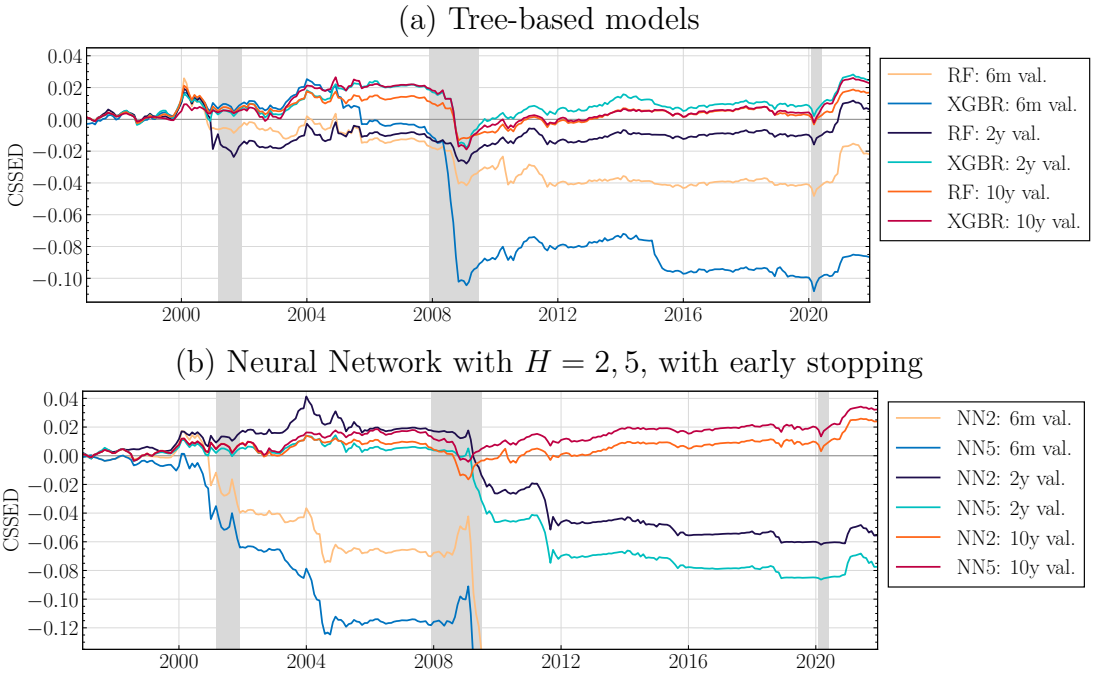
5.2 Breakdown in Forecasting Performance

Table 3 reports the sample-average forecasting performance for different models but does not provide any indication of how this evolves over time or whether forecasting performance breaks down in certain periods. To examine this issue, Figure 8 displays the cumulative sum of squared error differentials (CSSED) for our validated models against the zero prediction benchmark, with NBER recession periods marked by vertical grey shaded areas. Each panel

focuses on a different model: Panel (a) for Random Forest and Extreme Gradient Boosting, and Panel (b) for Neural Networks with 2 and 5 hidden layers.

Figure 8: Cumulative sum of squared error differentials

This figure displays Cumulative Sum of Squared Error Differentials (CSSED) with positive and increasing values indicating that the validated model is more accurate than the zero prediction benchmark. The data covers the test sample from January 1997 to December 2021. Grey shaded areas correspond to NBER recession periods. Panel (a) shows the CSSED of Random Forest and Extreme Gradient Boosting, while Panel (b) shows the performance for two Neural Networks, where the y-axis is capped at -0.11. Performance deteriorates until -0.35.



The forecasting performance of the tree-based models in Panel (a) deteriorates markedly during the three recessions in our sample, most notably during the GFC. This holds across all validation windows with the forecasts based on the six-month window experiencing the worst deterioration in performance during these periods. Panel (b) displays the performance of the shallow (NN2) and deep (NN5) neural network architectures, respectively. The worst performance is again observed for a 6-month validation window while a 10-year validation window produces better performance. The deeper architecture, NN5, shows slightly more variability in performance compared to NN2.[28]

[28]The CSSED for Lasso and Elastic Net are reported in Appendix Table C.2. Shorter validation windows

Overall, the performance observed across different ML models and validation windows exhibit similar patterns and no single model specification consistently surpasses the others throughout the entire test period.

5.3 Cross-sectional Variation in Predictive Accuracy

The top panel of Figure 9 reports percentiles of the cross-sectional distribution of $R^2_{i,OoS}$. For most models, the R^2_{OoS} is negative for between 25% and 50% of the stocks. The non-linear ML models generate a significantly greater cross-sectional spread in forecasting performance across stocks.

The bottom panel of Figure 9 illustrates how $R^2_{i,OoS}$ varies across firm size, captured by sorting stocks into $d = 10$ market-cap deciles. The smallest firms achieve the highest $R^2_{i,OoS}$ values across all models. These are generally also the firms that would have been most costly to trade, calling into question whether the return forecasts could have been exploited to generate economic gains. All models underperform for the largest firms, with $R^2_{i,OoS}$ values turning negative from the seventh decile and onward. The performance of the Neural Networks is, once again, particularly poor for the largest stocks.

While size is a proxy for how liquid a stock is, trading volume may provide an even better measure of how easy it would have been to convert forecasting signals from the ML models into trading positions. We therefore next examine results for portfolios of stocks sorted on trading volume. Table 4 shows strong evidence that stocks with the lowest trading volume also exhibit the strongest out-of-sample return predictability. Conversely, most methods tend to generate negative or very small positive R^2_{OoS} values for stocks in the top three volume-sorted deciles. This raises concerns about whether the return predictability identified by the ML methods could have been exploited for economic gains.

To evaluate this issue more formally, we report t-statistics as well as p-values for the Monotonic Relation (MR) test of Patton and Timmermann (2010) designed to test if there is a monotonically increasing relation between trading volume and return predictability.

help mitigate downturns during crises by turning to a zero prediction. The 2-year validation set typically yields the best results, while the 6-month window is the least accurate.

Figure 9: Out-of-sample performance across firms

Panel (a) displays firm-level $R^2_{i,OoS}$ values sorted into percentiles p while Panel (b) displays firm-level R^2_{OoS} values for stocks grouped into ten deciles based on their firm size (cut off at -1.6%. For $d = 10$, the performance deteriorates to -2% and -2.2% for NN2 and NN5, respectively). The validated models use a 2-year validation set. All measures are reported for firms with at least 60 observations in the test set, encompassing a total of 2774 firms. The test set ranges from January 1997 to December 2021.

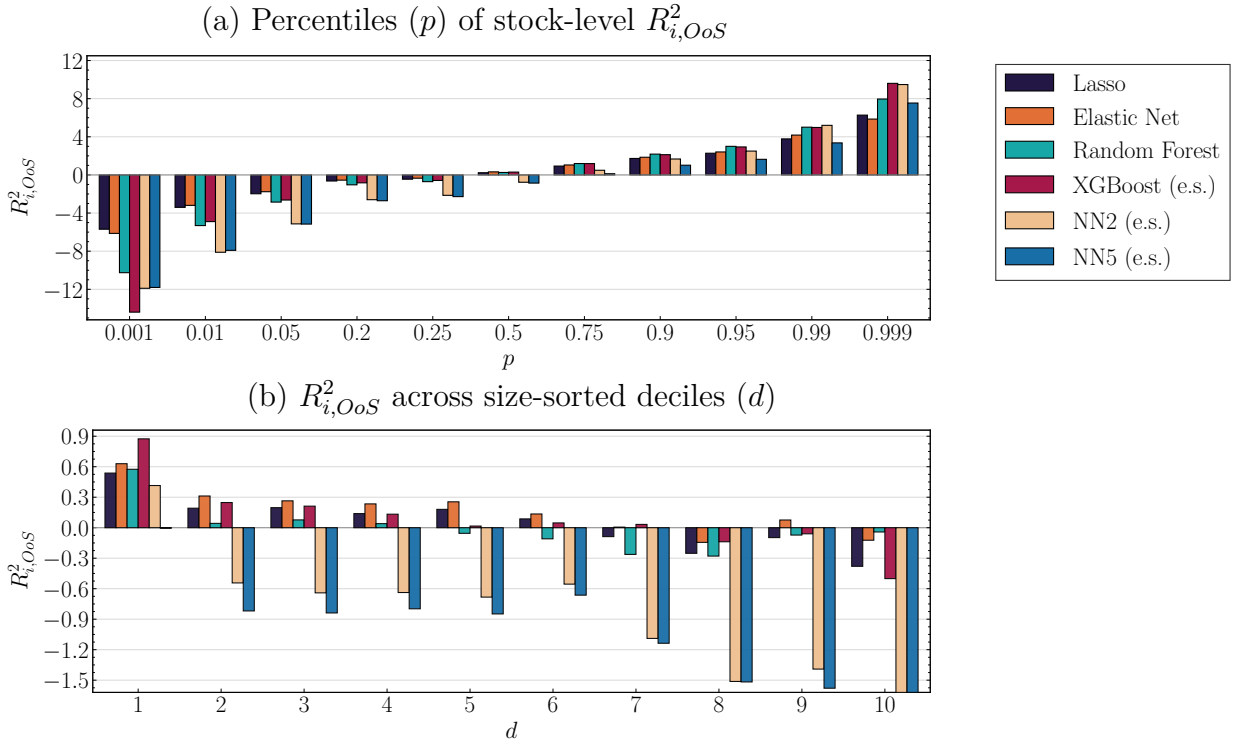


Table 4 shows that the opposite holds and that stocks with the highest trading volume tend to be significantly less predictable than stocks with lower trading volume. Combined with our results on the size-predictability relation, these findings suggest that converting the return forecasts from the ML models into profitable trading strategies, after accounting for the higher trading costs of small caps with low trading volume, would have been difficult or offered an unattractive risk-return trade-off.

Table 4: Out-of-sample R^2_{OoS} performance for portfolios sorted on trading volume

We report the R^2_{OoS} (monthly, percentage) forecasting performance for equal-weighted decile portfolios sorted on trading volume. We test the statistical significance of the H–L spread using HAC standard errors with **, and * denoting significance at the 1%, 5% levels, respectively. Additionally, we report the Patton and Timmermann (2010) monotonicity tests in the rows marked "MR Test", for which the null hypothesis is $H_0 : \mu_i \leq \mu_{i-1}$, where μ_i is the R^2_{OoS} value for the ith portfolio, versus the alternative $H_1 : \mu_i > \mu_{i-1}$. Hence, a rejection (low p-value) is consistent with monotonically increasing R^2_{OoS} values.

	Lasso			Enet			RF			XGBR		
	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y
Low	0.47	0.63	0.53	0.62	0.79	0.40	0.34	0.75	0.48	−0.11	0.48	0.70
2	0.09	0.32	0.37	0.21	0.46	0.25	−0.20	0.18	0.24	−3.52	0.33	0.42
3	0.18	0.35	0.30	0.29	0.48	0.20	0.05	0.33	0.39	−1.65	0.39	0.33
4	0.17	0.31	0.24	0.20	0.43	0.15	−0.24	0.12	0.28	−0.23	0.43	0.24
5	−0.14	0.07	0.12	−0.10	0.15	0.04	−0.28	−0.21	0.08	−0.49	0.18	0.15
6	0.17	0.26	0.20	0.13	0.37	0.10	−0.22	−0.13	0.13	−0.35	0.23	0.13
7	0.06	0.05	0.04	−0.01	0.11	−0.03	−0.19	−0.17	0.04	−0.25	0.11	0.07
8	−0.04	−0.06	0.01	−0.20	−0.01	−0.08	−0.55	−0.26	−0.01	−0.29	0.18	0.07
9	−0.15	−0.27	−0.09	−0.34	−0.17	−0.17	−0.73	−0.29	−0.25	−0.67	0.00	−0.03
High	0.00	−0.19	−0.01	−0.17	−0.12	−0.09	−0.38	−0.17	−0.08	−0.69	−0.08	−0.10
H–L	−0.47*	−0.82**	−0.54	−0.79**	−0.91**	−0.49	−0.73	−0.92	−0.56	−0.58	−0.56	−0.80*
MR Test (p-value)	0.95	0.783	0.296	0.78	0.754	0.287	0.815	0.722	0.569	0.307	0.481	0.41

	NN1			NN2			NN3			NN4			NN5		
	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y
Low	0.29	0.35	−1.66	0.29	0.44	−1.86	0.01	0.38	−1.84	−0.10	0.39	−2.34	−0.29	0.37	
2	−0.32	0.29	−2.54	−0.33	0.42	−2.64	−0.51	0.37	−2.56	−0.55	0.32	−2.96	−0.70	0.30	
3	−0.22	0.29	−2.48	−0.18	0.40	−2.62	−0.42	0.36	−2.58	−0.46	0.37	−3.01	−0.62	0.36	
4	−0.14	0.25	−2.34	−0.17	0.31	−2.48	−0.33	0.29	−2.43	−0.40	0.33	−2.79	−0.48	0.36	
5	−0.54	0.16	−3.02	−0.57	0.21	−3.06	−0.64	0.17	−2.98	−0.58	0.28	−3.37	−0.69	0.30	
6	−0.28	0.13	−2.94	−0.32	0.17	−3.15	−0.52	0.17	−3.13	−0.55	0.30	−3.54	−0.64	0.34	
7	−0.25	0.04	−2.30	−0.35	0.06	−2.42	−0.41	0.08	−2.40	−0.43	0.17	−2.64	−0.45	0.22	
8	−0.60	0.07	−3.36	−0.81	0.07	−3.51	−0.81	0.09	−3.45	−0.72	0.25	−3.79	−0.75	0.31	
9	−0.86	−0.08	−3.73	−1.08	−0.08	−3.98	−1.13	−0.08	−3.84	−1.01	0.10	−4.24	−1.08	0.15	
High	−0.52	0.00	−2.73	−0.82	−0.04	−2.82	−0.78	−0.02	−2.77	−0.66	0.11	−3.06	−0.66	0.17	
H–L	−0.81**	−0.35*	−1.06	−1.11**	−0.48**	−0.96	−0.79**	−0.39	−0.94**	−0.55**	−0.28**	−0.71**	−0.37**	−0.20	
MR Test (p-value)	0.71	0.78	0.25	0.88	0.69	0.33	0.84	0.49	0.30	0.80	0.70	0.38	0.82	0.67	0.60

5.4 Economic Value of Return Forecasts

We next examine the economic value of our return forecasts. We sort individual stocks into deciles based on their predicted returns for the following month and form equal-weighted portfolios. We then construct long-short portfolios taking a long positions in stocks in the top decile of expected returns and short positions in stocks in the bottom decile.[29]

Table 5 reports annualized average returns (Panel A) and Sharpe ratios (Panel B) of the prediction sorted portfolios. The high minus low (H–L) average return spread is significant

[29] Gu et al. (2020) and Avramov et al. (2023) follow this procedure to evaluate the economic significance of return forecasts. In contrast, Coulombe et al. (2022) develop a method based on Shapley values to estimate the contributions to portfolio performance of individual variables or groups of predictors.

across all models and validation sets except for the NN2 and NN5 models with 10-year validation sets, demonstrating that economic performance improves for stocks with the highest predicted return. However, we do not find any significant spreads for the Sharpe ratios.

We further evaluate whether the average returns of the decile portfolios sorted on return forecasts are monotonically increasing using the MR test of Patton and Timmermann (2010). Formally, we test the null of $H_0 : \mu_i \leq \mu_{i-1}$ against the alternative $H_1 : \mu_i > \mu_{i-1}$ so that rejections (small p-values) show that economic performance is higher for stocks with the highest predicted return while a failure to reject the null suggests the opposite. We reject the null for at most two validation windows across the Lasso, Enet, and RF methods (Panel A). In contrast, we fail to find significant evidence of a monotonically increasing relation for the portfolios based on the forecasts from the XGBR, NN2 and NN5 models.[30] Panel B of the table shows that the evidence is even weaker for the Sharpe ratios for which we only find a single case (Enet with a 2-year validation window) with a significantly increasing Sharpe ratio.

In conclusion, these results do not provide conclusive evidence that the return forecasts generated by applying validation to choose the hyperparameters of a range of ML methods could have been used to significantly improve economic performance.

[30]We also build 18 value-weighted portfolios using size, book to market, operating profitability, and investment as sorting variables. These sorts did not establish any particular patterns in return predictability across firm characteristics beyond the size effect.

Table 5: Economic performance of prediction-sorted stock portfolios

We report annualized average returns in Panel A and Sharpe ratios in Panel B for portfolios sorted on return predictions from six machine learning models: Lasso, Elastic Net (Enet), Random Forest (RF), Extreme Gradient Boosted Trees (XGBoost), and feed-forward neural networks with 2 and 5 hidden layers (NN2, NN5). The machine learning models are validated using windows of 6 months (6m), 2 years (2y), and 10 years (10y). We test the statistical significance of the H–L spread using HAC standard errors with **, and * denoting significance at the 1%, 5% levels, respectively. Additionally, we report the Patton and Timmermann (2010) monotonicity tests in the rows marked "MR Test (p-value)", for which the null hypothesis is $H_0 : \mu_i \leq \mu_{i-1}$, where μ_i is the mean of the return measure for the i^{th} portfolio, versus the alternative $H_1 : \mu_i > \mu_{i-1}$. Hence, a rejection (low p-value) is consistent with monotonically increasing mean returns or Sharpe ratios. All portfolios are equal-weighted.

Panel A: Average Returns

	Lasso			Enet			RF			XGBoost			NN2			NN5		
	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y
Low	10.44	8.46	12.59	10.65	8.29	13.03	10.53	11.65	10.66	12.26	13.12	13.22	12.58	10.53	13.24	16.25	13.79	14.75
2	9.01	9.51	9.70	12.17	6.78	9.53	12.25	13.83	9.61	14.60	6.98	22.07	10.10	11.39	13.69	−2.76	41.84	0.00
3	12.35	9.69	13.52	12.65	8.38	11.90	13.91	12.48	11.24	11.67	4.59	8.00	11.48	10.17	13.57	−8.89	51.43	0.00
4	10.69	10.97	11.98	13.50	9.65	13.08	13.19	11.36	12.83	7.45	2.14	4.30	11.36	12.31	9.01	5.06	39.13	0.00
5	11.78	10.75	13.36	15.25	8.90	13.14	14.74	11.95	12.78	19.66	7.00	4.42	11.24	11.68	13.25	−7.97	43.32	0.00
6	13.33	13.56	15.13	16.02	12.18	13.59	14.12	13.92	14.44	10.69	9.22	6.16	9.88	11.44	11.57	−9.21	47.61	0.00
7	16.90	16.87	13.90	17.81	12.44	14.92	14.58	10.72	15.46	11.56	8.97	23.61	12.66	11.39	12.30	−7.81	64.78	0.00
8	18.53	18.50	13.89	21.46	17.66	16.42	17.02	15.28	14.19	13.67	14.50	14.70	17.13	12.08	15.01	−16.47	63.39	0.00
9	16.96	19.39	19.12	21.93	18.54	17.11	17.78	15.29	17.46	12.28	18.72	27.20	12.86	14.60	15.91	−22.57	54.07	0.00
High	31.01	33.52	24.20	36.09	31.81	24.56	24.30	24.45	23.77	27.09	28.06	33.06	23.04	22.86	18.02	−31.07	50.34	−10.94
H–L	20.57**	25.06**	11.61**	25.44**	23.52**	11.53**	13.77**	12.80*	13.11**	14.83**	14.94**	19.84**	10.46*	12.33**	4.77	−47.32	36.55	−25.68
MR Test (p-value)	0.16	**0.01**	0.35	**0.00**	**0.05**	0.31	**0.03**	0.96	**0.04**	0.99	0.25	–	0.47	0.06	0.43	1.00	0.87	–

Panel B: Sharpe Ratios

	Lasso			Enet			RF			XGBoost			NN2			NN5		
	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y
Low	0.44	0.34	0.48	0.45	0.33	0.50	0.43	0.42	0.44	0.51	0.54	0.56	0.49	0.39	0.48	0.71	0.63	0.65
2	0.46	0.46	0.48	0.60	0.33	0.46	0.52	0.55	0.42	0.67	0.31	0.77	0.39	0.46	0.53	−0.12	1.94	0.00
3	0.63	0.50	0.68	0.64	0.43	0.59	0.59	0.51	0.53	0.51	0.20	0.30	0.47	0.43	0.57	−0.38	1.95	0.00
4	0.55	0.57	0.62	0.71	0.50	0.66	0.63	0.52	0.57	0.35	0.10	0.22	0.48	0.55	0.37	0.19	1.69	0.00
5	0.62	0.57	0.63	0.79	0.45	0.65	0.69	0.56	0.60	0.85	0.34	0.18	0.49	0.54	0.56	−0.34	1.79	0.00
6	0.62	0.67	0.72	0.78	0.60	0.63	0.66	0.66	0.66	0.46	0.50	0.29	0.42	0.52	0.50	−0.40	1.73	0.00
7	0.76	0.79	0.62	0.79	0.60	0.68	0.63	0.49	0.65	0.54	0.37	0.79	0.54	0.51	0.54	−0.27	2.16	0.00
8	0.74	0.76	0.62	0.85	0.75	0.73	0.72	0.60	0.59	0.50	0.59	0.58	0.68	0.53	0.64	−0.61	1.80	0.00
9	0.63	0.74	0.82	0.80	0.72	0.74	0.65	0.59	0.64	0.46	0.62	0.90	0.47	0.61	0.65	−0.83	1.72	0.00
High	0.76	0.98	0.82	0.90	0.96	0.85	0.72	0.75	0.70	0.77	0.79	0.83	0.72	0.79	0.59	−0.69	1.46	−0.41
H–L	0.73	1.14	0.77	0.79	1.19	0.74	0.58	0.50	0.57	0.71	0.80	0.71	0.65	0.67	0.27	−0.92	0.70	−1.19
MR Test (p-value)	0.60	0.56	0.90	0.21	**0.02**	0.34	0.79	0.67	0.72	0.81	0.17	–	0.64	0.46	0.37	0.93	0.87	–

6 Using the Bias-variance Trade-off to Guide the Search for Hyperparameters

Our empirical evidence shows that the predictive accuracy of ML algorithms is highly sensitive to the initial hyperparameter grid. In practice, specifying a reasonable hyperparameter space is challenging, particularly when dimensionality is high. To address this issue, we build on the insight that a small subset of hyperparameters are typically critical for determining out-of-sample performance. For example, in tree-based models, tree depth (D) and either the number of features (F) or the learning rate (η) are especially important. In neural networks, the learning rate and shrinkage parameter (α) play critical roles.

Recent studies increasingly rely on adaptive search algorithms, such as Hyperband, for hyperparameter tuning. We use Optuna (Akiba et al., 2019), an open-source hyperparameter optimization framework designed for efficient hyperparameter search in machine learning applications. Optuna implements the Tree-structured Parzen Estimator (TPE), which guides the search by splitting observed hyperparameter configurations into high and low performing groups, and prioritizing sampling from the more promising region. In addition, Optuna dynamically refines the search space as trials progress, allowing it to focus on more productive regions. This adaptivity improves the likelihood of identifying (near-)optimal configurations at lower computational cost.[31]

6.1 Narrow versus Wide Hyperparameter Spaces

We begin by evaluating the impact of hyperparameter space width on out-of-sample forecast performance. Table 6 reports results for Extreme Gradient Boosting (XGBoost) and a neural network with two hidden layers. For each model, we use Optuna to validate hyperparameters under three search space specifications: wide, baseline, and narrow. Across all validation windows (6 months, 2 years, 10 years), out-of-sample performance is highly sensitive to the range of the initial hyperparameter space. Wider hyperparameter grids consistently yield lower R^2_{OoS} values relative to the narrower alternatives.

[31]The combination of XGBoost and Optuna is often used in Kaggle forecasting competitions (see, e.g., Filho, 2023), and recently also by Coulombe et al. (2023).

These results highlight a key challenge in high-dimensional tuning. Overly broad search spaces increase the likelihood of exploring hyperparameter configurations that add little value, reducing out-of-sample performance.

Table 6: Performance statistics for the validated models using Optuna

We report the monthly R^2_{OoS} in percentage, and the corresponding Diebold-Mariano (DM) statistic for the validated models with different validation periods, where the MSE is used as the evaluation metric. "e.s." refers to early stopping. Results are based on 100 trials. "Baseline" grid corresponds to the initial search grids outlined in Table 2. For XGBoost, "Wide" grid is defined as $B = 1000$, $D \in [1, 12]$, $\eta \in [0.0001, 0.7]$, $\alpha \in [0, 0.5]$, $\lambda \in [0, 1]$, subsample and column sample rates in $[0, 1]$, and the loss function $\mathcal{L}(\cdot) \in \{\texttt{MSE}, \texttt{Huber}\}$. The "Narrow" grid uses $B = 1000$, $D \in [1, 2]$, $\eta \in [0.0001, 0.1]$, with early stopping and a patience of 5. For NN2, the "Wide" grid consists of $\eta \in [0.00001, 0.1]$, $\alpha \in [0.0001, 0.1]$, $\ell_2 \in [0.0001, 0.1]$, activation functions $f(\cdot) \in \{\texttt{Leaky}, \texttt{ReLU}, \texttt{Sigmoid}, \texttt{Linear}, \texttt{Tanh}, \texttt{Softmax}\}$, loss functions $\mathcal{L}(\cdot) \in \{\texttt{MSE}, \texttt{Huber}\}$, dropout rates in $\{0, 0.1, 0.2, 0.5\}$, and batch sizes of $\{2000, 10000, \texttt{none}\}$. The "Narrow" grid includes $\eta \in [0.0001, 0.001]$, $\alpha \in [0.0001, 0.01]$, activation functions $f(\cdot) \in \{\texttt{Leaky}, \texttt{ReLU}\}$, and loss functions $\mathcal{L}(\cdot) \in \{\texttt{MSE}, \texttt{Huber}\}$.

	Extreme Gradient Boosting (e.s.)					
	6 months		2 years		10 years	
Grid	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}
Wide	-1.911	-0.834	1.134	0.304	0.677	0.159
Baseline	-0.403	-0.161	0.641	0.227	0.774	0.217
Narrow	1.454	0.410	1.653	0.454	0.708	0.188
	Neural Networks $H = 2$ (e.s.)					
	6 months		2 years		10 years	
Grid	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}
Wide	-3.871	-3.001	-2.028	-0.818	1.080	0.245
Baseline	-3.430	-3.566	-1.722	-0.607	0.947	0.226
Narrow	0.165	0.015	0.575	0.174	1.239	0.287

6.2 Modeling the Joint Distribution of Hyperparameters

While the choice of hyperparameter ranges out-of-sample performance, our results suggest that incorporating structure into the search process can lead to further improvements. Figures 6 and 7 show that certain hyperparameters are *critical* for out-of-sample performance. For instance, shallow tree depths ($D \in \{1, 2\}$) and low learning rates ($\eta \in \{0.0001, 0.001, 0.01, 0.1\}$) yield robust performance in tree-based models. These hyperparameters interact in system-

atic ways. In tree-based models, robust out-of-sample performance requires shallow trees combined with low learning rates. In neural networks, improved out-of-sample performance require low learning rates and high regularization, achieved through larger shrinkage parameters (α). These patterns point to a structured relationship between critical hyperparameters, which we capture using joint priors.

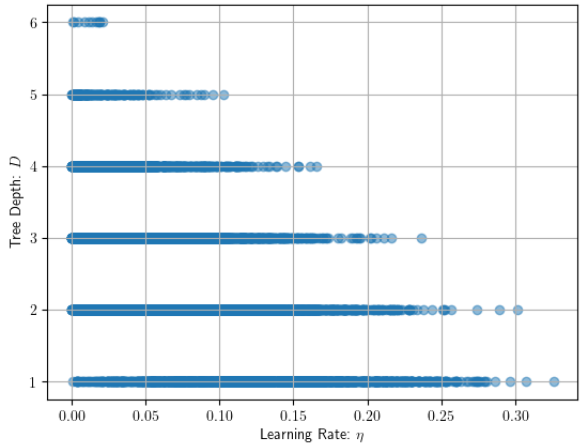
Following this discussion, we use a truncated bivariate normal distribution to implement our priors and Optuna to validate the hyperparameters. As in Table 2, we impose lower and upper bounds of $\underline{D} = 1$ and $\bar{D} = 6$ for tree depth (D) and lower and upper bounds of $\underline{\eta} = 0.0001$ and $\bar{\eta} = 0.4$ for the learning rate (η) for the XGBoost. We further assume a correlation of $\rho = -0.9$ between the learning rate and tree depth hyperparameters. The hyperparameters η and D follow a truncated normal with mean $\mu_\eta = \underline{\eta} + \kappa\Delta_\eta$ and $\mu_D = \underline{D} + \kappa\Delta_D$, where $\Delta_D = \bar{D} - \underline{D}$, $\Delta_\eta = \underline{\eta} - \bar{\eta}$. κ serves as a location parameter with lower values pulling the mean towards the lower bound. We choose the variance of the truncated normal distribution to have a 99% chance of falling below the upper bound. We follow the same procedure for neural networks, except we assume a positive correlation between the learning rate and shrinkage parameter ($\rho = 0.9$). Finally, we use the Gibbs sampler to generate 5000 draws from the bivariate truncated normal.

Figure 10 displays the empirical distributions of hyperparameter draws from the truncated normal priors for XGBoost (panel (a)) and the neural network (panel (b)). Table 7 reports the out-of-sample performance of both models under three search strategies: grid search, Optuna with uniform priors, and Optuna with truncated normal priors. For XGBoost, using truncated normal priors yields positive R^2_{OoS} values for all validation windows, highlighting the importance of jointly modeling learning rate and tree depth. In contrast, grid search and uniform prior search only produce positive R^2_{OoS} values under longer validation windows. For the neural networks, truncated priors with a 10-year validation window increases R^2_{OoS} by 0.8 and 0.5 percentage points relative to grid search and uniform priors, respectively. These findings demonstrate that structured hyperparameter search, guided by informative priors, can yield meaningful gains in out-of-sample accuracy. These improvements are most significant when standard approaches fail to identify well-performing configurations.

Figure 10: Empirical Distribution of the Truncated Normal Priors

This figure shows the empirical distribution of hyperparameter draws sampled by Optuna under bivariate truncated normal priors. Panel (a) displays the XGBoost hyperparameters, with a correlation of $\rho = -0.9$ between tree depth and learning rate. Panel (b) shows the neural network hyperparameters, where a positive correlation of $\rho = 0.9$ is imposed between the learning rate and shrinkage parameter.

(a) Extreme Gradient Boosting



(b) Neural Networks

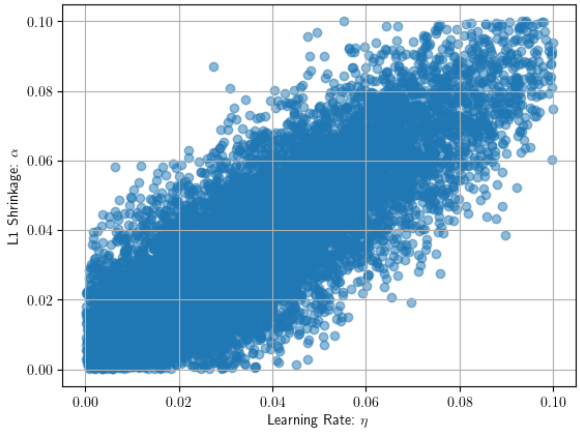


Table 7: Out of sample performance comparison of validated models

This table reports monthly R^2_{OoS} values in percentage points and corresponding Diebold-Mariano (DM) statistic for the validated models across different validation periods and hyperparameter search methods. Panel A presents results for the XGBoost model under three search strategies: grid search, Optuna with uniform priors, Optuna with truncated normal priors (with negative correlation). Panel B presents results for the neural networks model under three search strategies: grid search, Optuna with uniform priors, Optuna with truncated normal priors (with positive correlation).

	Panel A: Extreme Gradient Boosting (e.s.)					
	Grid Search		Uniform Priors		Trunc. Normal Priors	
Val.	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}
6m	-1.398	-0.795	-0.403	-0.161	0.393	0.116
2y	0.648	0.221	0.641	0.227	0.687	0.200
10y	0.636	0.197	0.774	0.217	0.343	0.089
	Panel B: Neural Networks $H = 2$ (e.s.)					
	Grid Search		Uniform Priors		Trunc. Normal Priors	
Val.	DM	R^2_{OoS}	DM	R^2_{OoS}	DM	R^2_{OoS}
6m	-3.294	-2.753	-3.430	-3.566	-3.726	-3.400
2y	-1.310	0.451	-1.722	-0.607	-1.953	-0.723
10y	0.785	0.195	0.947	0.226	1.165	0.275

7 Conclusion

Machine learning models such as random forests, gradient boosted trees, and neural networks, have shown considerable promise in empirical asset pricing studies. This capacity, however, comes with the challenge of managing a broad array of hyperparameters—from the number of hidden layers in neural networks to the depth of trees in random forests—each impacting model performance. These hyperparameters determine the ML models' basic architecture and how they regulate the bias-variance trade-off that is key to forecasting performance especially for variables as noisy as stock returns.

The risk of overfitting and spurious forecasting performance is not particular to ML methods and has been a constant theme in evaluating the forecasting performance even of linear models. However, ML methods display a degree of flexibility that significantly heightens this danger. Our empirical analysis analysis shows that the distribution of out-of-sample forecasting performance is much wider and highly left-skewed for ML methods compared to linear models. Risk related to model misspecification and parameter estimation error is, thus, notably elevated for flexible ML models that attempt to account for nonlinear dynamics and complex interaction terms. This flexibility can come at the cost of enhanced sensitivity to noisy data and outliers.

Our extensive analysis of the importance of hyperparameters for ML models' forecasting performance shows that there is a ("guaranteed success") range of values for a subset of one or two hyperparameters which results in impressive out-of-sample forecasting performance regardless of how the other hyperparameters are chosen. Conversely, there is also a ("lost cause") range of values for a subset of hyperparameters for which poor out-of-sample forecasting performance occurs regardless of how the other hyperparameters are chosen.

Importantly, time-series validation schemes which use data-driven methods to choose the optimal combination of hyperparameters do not provide a definitive solution to the dependence of forecasting performance on the chosen hyperparameters. Validation methods themselves depend on features such as the choice of validation window and the grid of hyperparameter values included in the search and these choices tend to have important effects on forecasting performance.

Our results highlight that the current lack of guidelines for setting the range of values examined for the hyperparameters heightens the risk of false inference in asset pricing studies based on out-of-sample forecasting performance. An implication of our findings is that empirical studies should carefully document the range of hyperparameters being considered along with details of any validation schemes being used. Further, we show that improvements in predictive accuracy are possible by considering the joint effect of the values of hyperparameters that control the bias-variance trade-off of the machine learning forecasting methods.

References

Akiba, T., S. Sano, T. Yanase, T. Ohta, and M. Koyama (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631.

Amari, S., N. Murata, K.-R. Müller, M. Finke, and H. Yang (1995). Statistical theory of overtraining—is cross-validation asymptotically effective? *Advances in Neural Information Processing Systems 8*.

Avramov, D., S. Cheng, and L. Metzker (2023). Machine learning vs. economic restrictions: Evidence from stock return predictability. *Management Science 69*(5), 2587–2619.

Bali, T. G., H. Beckmeyer, M. Moerke, and F. Weigert (2023). Option return predictability with machine learning and big data. *Review of Financial Studies 36*(9), 3548–3602.

Bergstra, J. and Y. Bengio (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13*(2), 282–305.

Bianchi, D., M. Büchner, and A. Tamoni (2021). Bond risk premiums with machine learning. *Review of Financial Studies 34*(2), 1046–1089.

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

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Trees*. CRC Press.

Bryzgalova, S., S. Lerner, M. Lettau, and M. Pelger (2024). Missing financial data. *Review of Financial Studies (forthcoming) hhae036*, 1–54.

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

Cao, S., W. Jiang, J. Wang, and B. Yang (2024). From man vs. machine to man + machine: The art and AI of stock analyses. *Journal of Financial Economics 160*, 103910.

Chen, L., M. Pelger, and J. Zhu (2024). Deep learning in asset pricing. *Management Science 70*(2), 714–750.

Chen, T. and C. Guestrin (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 785–794.

Coulombe, P. G., M. Leroux, D. Stevanovic, and S. Surprenant (2022). How is machine learning useful for macroeconomic forecasting? *Journal of Applied Econometrics 37*(5), 920–964.

Coulombe, P. G., D. E. Rapach, E. C. M. Schütte, and S. Schwenk-Nebbe (2023). The anatomy of machine learning-based portfolio performance. Working Paper.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems 2*(4), 303–314.

Diebold, F. X. and R. S. Mariano (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics 13*(3), 253–263.

Filho, M. (2023). Xgboost hyperparameter tuning with optuna. `https://forecastegy.com/posts/xgboost-hyperparameter-tuning-with-optuna/`. Accessed on: October 8, 2023.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29*(5), 1189–1232.

Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings.

Green, J., J. R. Hand, and X. F. Zhang (2017). The characteristics that provide independent information about average US monthly stock returns. *Review of Financial Studies 30*(12), 4389–4436.

Gu, S., B. Kelly, and D. Xiu (2020). Empirical asset pricing via machine learning. *Review of Financial Studies 33*(5), 2223–2273.

Gu, S., B. Kelly, and D. Xiu (2021). Autoencoder asset pricing models. *Journal of Econometrics 222*(1), 429–450.

Han, Y., A. He, D. Rapach, and G. Zhou (2023). Cross-sectional expected returns: New Fama-MacBeth regressions in the era of machine learning. Working paper.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366.

Jiang, J., B. Kelly, and D. Xiu (2023). (re-)imag(in)ing price trends. *Journal of Finance 78*(6), 3193–3249.

Kelly, B., D. Xiu, et al. (2023). Financial machine learning. *Foundations and Trends® in Finance 13*(3-4), 205–363.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, Volume 114, pp. 953–956. Russian Academy of Sciences.

Masini, R. P., M. C. Medeiros, and E. F. Mendes (2023). Machine learning advances for time series forecasting. *Journal of Economic Surveys 37*(1), 76–111.

Menkveld, A. J., A. Dreber, F. Holzmeister, J. Huber, M. Johannesson, M. Kirchler, S. Neusüss, M. Razen, U. Weitzel, D. Abad-Díaz, et al. (2024). Nonstandard errors. *Journal of Finance 79*(3), 2339–2390.

Patton, A. J. and A. Timmermann (2010). Monotonicity in asset returns: New tests with applications to the term structure, the CAPM, and portfolio sorts. *Journal of Financial Economics 98*(3), 605–625.

Prechelt, L. (2002). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pp. 55–69. Springer.

Probst, P., M. N. Wright, and A.-L. Boulesteix (2019). Hyperparameters and tuning strategies for random forest. *WIREs Data Mining Knowl Discov 9*, e1301.

Shen, Z. and D. Xiu (2024). Can machines learn weak signals? Working Paper.

Van Binsbergen, J. H., X. Han, and A. Lopez-Lira (2023). Man versus machine learning: The term structure of earnings expectations and conditional biases. *Review of Financial Studies 36*(6), 2361–2396.

Wolff, D. and U. Neugebauer (2019). Tree-based machine learning approaches for equity market predictions. *Journal of Asset Management 20*(4), 273–288.

Appendix for

"Overhyped? Can ML Models Reliably Predict Stock Returns?"

S. Yanki Kalfa, Allan Timmermann, Terri van der Zwan

This supplementary material for "Overhyped? Can ML Models Reliably Predict Stock Returns?" complements the paper with details on the data and additional empirical results.

A Data

Table A.1 reports details on the firm characteristics used in our paper. Table A.2 presents the categorization of our dataset based on the Standard Industry Classification system used to generate industry dummies. For each grey block in the table, a separate dummy is constructed, generating a total of 48 distinct dummy variables.

Table A.1: Details of the 92 firm characteristics

No.	Acronym	Firm characteristic description	Update frequency
1	absacc	Absolute accruals	Annual
2	acc	Working capital accruals	Annual
3	aeavol	Abnormal earnings announcement volume	Quarterly
4	age	No. of years since first Compustat coverage	Annual
5	agr	Asset growth	Annual
6	baspread	Bid-ask spread	Monthly
7	beta	Market beta	Monthly
8	betasq	Market beta squared	Monthly
9	bm	Book-to-market	Annual
10	bm_ia	Industry-adjusted book to market	Annual
11	cash	Cash holdings	Quarterly
12	cashdebt	Cash flow to debt	Annual
13	cashpr	Cash productivity	Annual
14	cfp	Cash flow to price ratio	Annual
15	cfp_ia	Industry-adjusted cash flow to price ratio	Annual
16	chatoia	Industry-adjusted change in asset turnover	Annual
17	chcsho	Change in shares outstanding	Annual
18	chempia	Industry-adjusted change in employees	Annual
19	chinv	Change in inventory	Annual
20	chmom	Change in 6-month momentum	Monthly
21	chpmia	Industry-adjusted change in profit margin	Annual
22	chtx	Change in tax expense	Quarterly
23	cinvest	Corporate investment	Quarterly
24	convind	Convertible debt indicator	Annual
25	currat	Current ratio	Annual
26	depr	Depreciation / PP&E	Annual
27	divi	Dividend initiation	Annual
28	divo	Dividend omission	Annual
29	dolvol	Dollar trading volume	Monthly
30	dy	Dividend to price	Annual

Table A.1—*continued*

No.	Acronym	Firm characteristic description	Update frequency
31	ear	Earnings announcement return	Quarterly
32	egr	Growth in common shareholder equity	Annual
33	ep	Earnings to price	Annual
34	gma	Gross profitability	Annual
35	grcapx	Growth in capital expenditures	Annual
36	grltnoa	Growth in long term net operating assets	Annual
37	herf	Industry sales concentration	Annual
38	hire	Employee growth rate	Annual
39	idiovol	Idiosyncratic return volatility	Monthly
40	ill	Illiquidity	Monthly
41	indmom	Industry momentum	Monthly
42	invest	Capital expenditures and inventory	Annual
43	lev	Leverage	Annual
44	lgr	Growth in long-term debt	Annual
45	maxret	Maximum daily return	Monthly
46	mom12m	12-month momentum	Monthly
47	mom1m	1-month momentum	Monthly
48	mom36m	36-month momentum	Monthly
49	mom6m	6-month momentum	Monthly
50	ms	Financial statement score	Quarterly
51	mvel1	Size, market capitalization	Monthly
52	mve_ia	Industry-adjusted size	Annual
53	nincr	Number of earnings increases	Quarterly
54	operprof	Operating profitability	Annual
55	orgcap	Organizational capital	Annual
56	pchcapx_ia	Industry adjusted % change in capital expenditures	Annual
57	pchcurrat	% change in current ratio	Annual
58	pchdepr	% change in depreciation	Annual
59	pchgm_pchsale	% change in gross margin − % change in sales	Annual
60	pchquick	% change in quick ratio	Annual
61	pchsale_pchinvt	% change in sales − % change in inventory	Annual
62	pchsale_pchrect	% change in sales − % change in A/R	Annual
63	pchsale_pchxsga	% change in sales − % change in SG&A	Annual
64	pchsaleinv	% change sales-to-inventory	Annual
65	pctacc	Percent accruals	Annual
66	pricedelay	Price delay	Monthly
67	ps	Financial statements score	Annual
68	quick	Quick ratio	Annual
69	rd	R&D increase	Annual
70	rd_mve	R&D to market capitalization	Annual
71	rd_sale	R&D to sales	Annual
72	retvol	Return volatility	Monthly
73	roaq	Return on assets	Quarterly
74	roavol	Earnings volatility	Quarterly
75	roeq	Return on equity	Quarterly
76	roic	Return on invested capital	Annual
77	rsup	Revenue surprise	Quarterly
78	salecash	Sales to cash	Annual
79	saleinv	Sales to inventory	Annual
80	salerec	Sales to receivables	Annual
81	securedind	Secured debt indicator	Annual
82	sgr	Sales growth	Annual
83	sin	Sin stocks (smoke, tobacco, beer, alcohol, or gaming)	Annual
84	sp	Sales to price	Annual
85	std_dolvol	Volatility of liquidity (dollar trading volume)	Monthly
86	std_turn	Volatility of liquidity (share turnover)	Monthly
87	stdacc	Accrual volatility	Quarterly
88	stdcf	Cash flow volatility	Quarterly
89	tang	Debt capacity/firm tangibility	Annual
90	tb	Tax income to book income	Annual
91	turn	Share turnover	Monthly
92	zerotrade	Zero trading days	Monthly

Note: 2 characteristics of Gu et al. (2020) are not considered; `realestate` and `secured`, no. 72, 82 in their paper, respectively. For the authors and journals corresponding to the anomalies, see Green et al. (2017).

Table A.2: Grouping of Standard Industry Classification codes

SIC2 digits	Industry
1	Agriculture, Forestry, and Fishing
2	Mining
7	Construction
10	Manufacturing
12	Transportation, Communications, Electric, Gas, and Sanitary Services
13	Wholesale Trade
14	Retail Trade
15	Eating and Drinking Places
16	Miscellaneous Retail
17	Finance, Insurance, and Real Estate
20	Services
21	Miscellaneous Services
22	Public Administration
23	Non-Classifiable Establishments
24	Educational Services
25	Health Services
26	Legal Services
27	Engineering, Accounting, Research, Management, and Related Services
28	Agricultural Services
29	Miscellaneous Services
30	Apparel and Other Finished Products Made From Fabrics and Similar Materials
31	Leather and Leather Products
32	Stone, Clay, Glass, and Concrete Products
33	Primary Metal Industries
34	Fabricated Metal Products, Except Machinery and Transportation Equipment
35	Industrial and Commercial Machinery and Computer Equipment
36	Electronic and Other Electrical Equipment and Components, Except Computer Equipment
37	Transportation Equipment
38	Instruments and Related Products
39	Miscellaneous Manufacturing Industries
40	Railroad Transportation
42	Motor Freight Transportation and Warehousing
44	Water Transportation
45	Transportation by Air
46	Pipelines, Except Natural Gas
47	Transportation Services
48	Communications
49	Electric, Gas, and Sanitary Services
50	Wholesale Trade-Durable Goods
51	Wholesale Trade-Non-Durable Goods
52	Building Materials, Hardware, Garden Supply, and Mobile Home Dealers
53	General Merchandise Stores
54	Food Stores
55	Automotive Dealers and Gasoline Service Stations
56	Apparel and Accessory Stores
57	Home Furniture, Furnishings, and Equipment Stores
58	Eating and Drinking Places
59	Miscellaneous Retail Stores
60	Depository Institutions
61	Non-depository Credit Institutions
62	Security and Commodity Brokers, Dealers, Exchanges, and Services
63	Insurance Carriers
64	Insurance Agents, Brokers, and Service
65	Real Estate
67	Holding and Other Investment Offices
70	Hotels, Rooming Houses, Camps, and Other Lodging Places
72	Personal Services
73	Business Services
75	Automotive Repair, Services, and Parking
76	Miscellaneous Repair Services
78	Motion Pictures
79	Amusement and Recreation Services
80	Health Services
82	Educational Services
83	Social Services
87	Engineering, Accounting, Research, Management, and Related Services
99	Non-Classifiable Establishments

Note: Rows that are marked grey are grouped.

B Details of the Algorithms

Algorithm 1 shows the early stopping procedure. Early stopping is used by extreme gradient boosting and neural networks.

Algorithm 1 *Early Stopping*

Require: Patience parameter ρ and initial model parameters θ_M.
  Set prediction error $e = \infty$, and counter $p = 0$.
  while $p < \rho$ **do**
    Obtain parameters $\widehat{\theta}_m$ from ML procedure.
    Calculate prediction error $\widehat{e}$ in the validation sample.
    if $\widehat{e} < e$ **then**
      $p \leftarrow 0$ ▷ Reset counter
      $e \leftarrow \widehat{e}$ ▷ Update prediction error
      $\theta_M \leftarrow \widehat{\theta}_M$ ▷ Update model parameters
    else
      $p \leftarrow p + 1$ ▷ Update counter
    end if
  end while
  return θ_M ▷ Resulting model parameter

Algorithm 2 below shows the Adam optimizer with ω containing both the weights and biases in the neural network.

Algorithm 2 *Adam Optimizer*

Require: Learning rate η, exponential decay rates β_1, β_2, objective function $\mathcal{L}(\omega)$, and initial parameter vector ω_0.
  Set first moment $m_0 = 0$, second moment $v_0 = 0$, and step size $d = 0$.
  while ω_d not converged **do**
    $d \leftarrow d + 1$
    $g_d \leftarrow \nabla_\omega \mathcal{L}_d(\omega_{d-1})$
    $m_d \leftarrow \beta_1 \cdot m_{d-1} + (1 - \beta_1) \cdot g_d$ ▷ Update biased first moment
    $v_d \leftarrow \beta_2 v_{d-1} + (1 - \beta_2) g_d^2$ ▷ Update biased second moment
    $\hat{m}_d \leftarrow m_d / (1 - \beta_1^d)$ ▷ Bias corrected first moment
    $\hat{v}_d \leftarrow v_d / (1 - \beta_2^d)$ ▷ Bias corrected second moment
    $\omega_d \leftarrow \omega_{d-1} - \eta \cdot \hat{m}_d / (\sqrt{\hat{v}_d} + \epsilon)$ ▷ Update parameters
  end while
  return ω_d ▷ Resulting parameter

C Additional Empirical Results

C.1 Dynamic Adjustments in the Selected Hyperparameters

To gain insight into how the validation method adapts to the data over time, Figure C.1 shows the sequence of hyperparameter values selected over time for the Lasso, Elastic Net, Random Forest, XGBoost, and Neural Networks with 2 and 5 hidden layers. In each figure, the shaded areas represent the full grid available for hyperparameter selection.

For the Lasso approach, panel (a), the hyperparameter α generally falls below 0.01. Approximately 5% of the regression coefficients (β) remain non-zero in most windows, equating to seven distinct features. When $\alpha = 0.012$, the model produces zero predictions. Furthermore, the validated models select a maximum of 18 features, highlighting the amount of shrinkage used.

For the Elastic Net approach, panel (b), the range of validated hyperparameter values is wider with a maximum of $\alpha = 0.015$ while λ fluctuates over the full range between 0.2 and 0.8. Moreover, values for the two hyperparameters are negatively correlated so that when λ is high, α tends to be low. When the model selects fewer predictors (high α), it applies less shrinkage of their coefficients while larger models require more shrinkage of the individual coefficients.

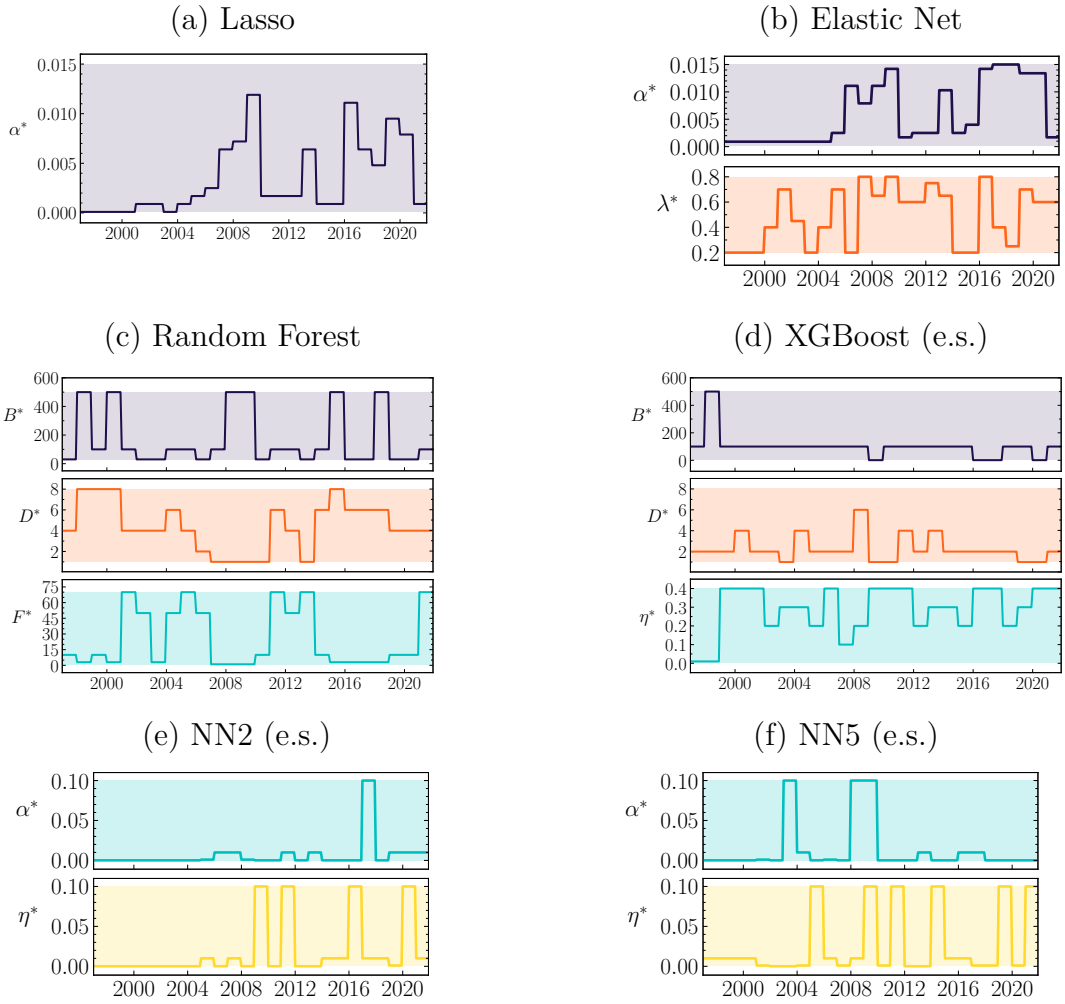
Panel (c) shows the sequence of hyperparameter values selected by the validation procedure applied to the Random Forest. The number of trees (B^*) varies, and there is often a trade-off between the depth and number of features, meaning that when D^* is high, F^* is often low and vice versa.

For XGBoost, panel (d), the validation procedure tends to favor larger learning rates, leading to overfitting as shown in Table 3. However, the validation procedure tends to pick shallow trees with a depth ($D^* = 2$). Combined with the learning rate η^*, the XGBoost prefers relatively large learning rates with shallow trees to guard against over-fitting.

Panels (e) and (f) show the sequence of the selected hyperparameters for the neural networks. For each architecture, the selected hyperparameters are displayed as the median over the five neural networks with distinct random seeds. A small learning rate, η, between 0.0001 and 0.001, is usually chosen. This pattern does not apply uniformly across models validated on the test set, however. The chosen learning rate, η^*, can vary a great deal and sometimes a higher rate of 0.01 is preferred. For the 2-year validation period the network weights require less shrinkage as indicated by a lower α value. Typically, a higher level of shrinkage is associated with a reduced learning rate. This relationship suggests a trade-off between the rate of learning and the degree of regularization applied to the model, with adjustments to one parameter potentially necessitating compensatory changes to another.

Figure C.1: Sequence of hyperparameters selected by the validated models

The Figure displays the sequence of hyperparameters chosen based on a 2-year validation period. The models are refitted every 12 months which explains the steps. Shaded regions in the figure represent the conventional grids for hyperparameter selection.



C.2 Correlations between Predictions across Validated Models

Table C.1 shows average stock-level correlation between return predictions from the individual validated model using 6-month, 2-year and 10-year validation sets. In general, correlations tend to be quite low both when comparing return predictions from different models as well as return predictions generated by a particular model but using different validation sets.

Table C.1: Correlations between predictions across validated models

		Lasso			Elastic Net			Random Forest			XGBoost			NN2			NN5		
		6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y	6m	2y	10y
Lasso	6m	1.00	0.64	0.50	0.90	0.63	0.49	0.42	0.42	0.42	0.19	0.25	0.19	0.57	0.48	0.31	0.50	0.36	0.30
	2y	0.64	1.00	0.42	0.74	0.95	0.46	0.34	0.33	0.39	0.12	0.17	0.12	0.45	0.64	0.27	0.38	0.39	0.17
	10y	0.50	0.42	1.00	0.55	0.54	0.96	0.35	0.36	0.42	0.22	0.29	0.31	0.30	0.36	0.55	0.24	0.17	0.17
Enet	6m	0.90	0.74	0.55	1.00	0.74	0.58	0.46	0.43	0.45	0.21	0.28	0.18	0.60	0.53	0.34	0.51	0.35	0.28
	2y	0.63	0.95	0.54	0.74	1.00	0.57	0.37	0.38	0.44	0.15	0.21	0.15	0.48	0.66	0.32	0.41	0.38	0.17
	10y	0.49	0.46	0.96	0.58	0.57	1.00	0.39	0.36	0.47	0.26	0.33	0.35	0.30	0.37	0.57	0.24	0.18	0.18
RF	6m	0.42	0.34	0.35	0.46	0.37	0.39	1.00	0.70	0.56	0.48	0.44	0.39	0.38	0.33	0.23	0.35	0.27	0.19
	2y	0.42	0.33	0.36	0.43	0.38	0.36	0.70	1.00	0.53	0.38	0.45	0.43	0.35	0.35	0.16	0.34	0.30	0.17
	10y	0.42	0.39	0.42	0.45	0.44	0.47	0.56	0.53	1.00	0.38	0.41	0.46	0.42	0.40	0.20	0.42	0.37	0.27
XGBR	6m	0.19	0.12	0.22	0.21	0.15	0.26	0.48	0.38	0.38	1.00	0.55	0.29	0.09	0.05	0.17	0.05	-0.03	0.08
	2y	0.25	0.17	0.29	0.28	0.21	0.33	0.44	0.45	0.41	0.55	1.00	0.32	0.09	0.08	0.17	0.06	-0.04	0.11
	10y	0.19	0.12	0.31	0.18	0.15	0.35	0.39	0.43	0.46	0.29	0.32	1.00	0.12	0.18	0.15	0.12	0.14	0.03
NN2	6m	0.57	0.45	0.30	0.60	0.48	0.30	0.38	0.35	0.42	0.09	0.09	0.12	1.00	0.57	0.09	0.96	0.59	0.16
	2y	0.48	0.64	0.36	0.53	0.66	0.37	0.33	0.35	0.40	0.05	0.08	0.18	0.57	1.00	0.22	0.52	0.78	0.37
	10y	0.31	0.27	0.55	0.34	0.32	0.57	0.23	0.16	0.20	0.17	0.17	0.15	0.09	0.22	1.00	0.03	0.06	0.27
NN5	6m	0.50	0.38	0.24	0.51	0.41	0.24	0.35	0.34	0.42	0.05	0.06	0.12	0.96	0.52	0.03	1.00	0.64	0.20
	2y	0.36	0.39	0.17	0.35	0.38	0.18	0.27	0.30	0.37	-0.03	-0.04	0.14	0.59	0.78	0.06	0.64	1.00	0.46
	10y	0.30	0.17	0.17	0.28	0.17	0.18	0.19	0.17	0.27	0.08	0.11	0.03	0.16	0.37	0.27	0.20	0.46	1.00

Note: The average over the time series correlations between predictions from different models for each firm with over 60 observations in the test sample.

C.3 Evolution in Forecasting Performance for the Linear Models

Figure C.2 shows the cumulative sum of squared error differentials for the linear models across different validation windows. For these linear models, a shorter validation periods helps mitigate downturns during crises.

Figure C.2: Cumulative sum of squared error differentials

This figure displays the Cumulative Sum of Squared Error Differentials (CSSED) for validated linear Lasso and Elastic Net models compared to a zero prediction benchmark. The data covers the sample period from 1997 to 2021. Grey shaded areas correspond to NBER recession periods.

