A machine learning approach to volatility forecasting*

Kim Christensen† Mathias Siggaard† Bezirgen Veliyev†

January 13, 2021

Abstract

We show that machine learning (ML) algorithms improve one-day-ahead forecasts of realized variance from 29 Dow Jones Industrial Average index stocks over the sample period 2001 – 2017. We inspect several ML approaches: Regularization, tree-based algorithms, and neural networks. Off-the-shelf ML implementations beat the Heterogeneous AutoRegressive (HAR) model, even when the only predictors employed are the daily, weekly, and monthly lag of realized variance. Moreover, ML algorithms are capable of extracting substantial more information from additional predictors of volatility, including firm-specific characteristics and macroeconomic indicators, relative to an extended HAR model (HAR-X). ML automatically deciphers the often nonlinear relationship among the variables, allowing to identify key associations driving volatility. With accumulated local effect (ALE) plots we show there is a general agreement about the set of the most dominant predictors, but disagreement on their ranking. We investigate the robustness of ML when a large number of irrelevant variables, exhibiting serial correlation and conditional heteroscedasticity, are added to the information set. We document sustained forecasting improvements also in this setting.

JEL Classification: C10; C50.

Keywords: Gradient boosting, high-frequency data, machine learning, neural network, random forest, realized variance, regularization, volatility forecasting.

*Our work was supported by CREATES. The paper was presented at the 13th International Conference on Computational and Financial Econometrics (CFE 2019) in London, UK. We thank Torben Andersen and Tim Bollerslev for their extensive comments and insightful suggestions that helped to improve many aspects of this paper. Please address correspondence to: siggaard@econ.au.dk.

†CREATES, Department of Economics and Business Economics, Aarhus University, Fuglesangs Allé 4, 8210 Aarhus V, Denmark.

1 Introduction

The unprecedented volatility in financial markets during the last decade has led to increased awareness about the importance of explaining key drivers behind such movements. On the one hand, the structure of financial markets is complex, highly nonlinear and has a low signal-to-noise ratio, which makes it nearly impossible to predict short-term asset returns (see, e.g., Gu, Kelly, and Xiu, 2020; Chen, Pelger, and Zhu, 2019). On the other hand, various stylized facts (e.g., the slow decay of autocorrelation in absolute returns), implies that volatility is to a large extent predictable. With its crucial role in asset pricing, portfolio allocation, and risk management, volatility forecasting has therefore attracted a large amount of attention.

The origins of this literature go back to Engle (1982); Bollerslev (1986); Taylor (1982), who developed discrete-time GARCH and stochastic volatility processes for modeling autoregressive conditional heteroskedasticity (see Hansen and Lunde, 2005, for a broader review). As argued by Corsi (2009), however, standard GARCH and stochastic volatility models are unable to reproduce all of the salient features of financial markets. He proposes the nonparametric Heterogeneous Auto-Regressive (HAR) model, which combines volatility components at different frequencies to capture heterogeneous types of market participants. Although the HAR model is highly parsimonious, the complicated structure of the underlying data renders it inadequate in some directions. Therefore, a number of extensions of the baseline HAR have been proposed (e.g., Andersen, Bollerslev, and Diebold, 2007; Corsi and Renò, 2012; Bollerslev, Patton, and Quaedvlieg, 2016). These are solely based on past return histories as conditioning information, which is surprising, since a large amount of research documents an intimate relationship between news announcements and volatility (e.g., Schwert, 1989; Engle, Ghysels, and Sohn, 2013; Bollerslev, Li, and Xue, 2018).

The reason no or only a small subset of additional covariates are included in volatility prediction is due to the fact that traditional models, often relying on linear regression, break down when the explanatory variables are strongly correlated, exhibit low signal-to-noise ratio, or if the underlying structure is highly nonlinear. As explained above, this is a trademark of the equity market, so it is unsurprising that relatively few studies conduct volatility forecasting in a data-rich environment. The recent access to large datasets therefore highlights a need for investigating more powerful tools that take these problems into account.

As discussed in Varian (2014), machine learning (ML) techniques possess these capabilities. Up to now, however, only a few studies combine ML and volatility forecasting using a single approach, e.g., Audrino and Knaus (2016); Audrino, Sigrist, and Ballinari (2020); Caporin and Poli (2017) investigate lasso, Luong and Dokuchaev (2018) use random forests, Mittnik, Robinzonov, and Spindler (2015) apply component-wise gradient boosting, Bucci (2020); Donaldson and Kamstra (1997); Hillebrand and Medeiros (2010); Fernandes, Medeiros, and Scharth (2014); Rahimikia and Poon (2020) explore neural networks. More recent work includes Carr, Wu, and Zhang (2019), who study the application of ML to low-frequency options data in order to forecast 30-day realized variance of the S&P 500 index.

In contrast to previous studies, in this paper we conduct a comprehensive analysis of the out-of-

sample performance of multiple ML techniques for volatility forecasting, i.e. regularization (ridge, lasso, elastic net), tree-based algorithms (bagging, random forest, gradient boosting), and neural networks. We compare these to the HAR model. We aim not only to do a complete comparison across traditional and ML methods but also provide evidence of how and why some of these methods improve the accuracy of forecasting volatility.

The findings of this paper are five-fold. First, we show superior out-of-sample forecasting accuracy with off-the-shelf implementations of various ML techniques compared to the HAR model. Second, we find that substantial incremental information about future volatility can be extracted with ML from additional volatility predictors with minimal noise fitting, if regularization is applied to account for overfitting. In contrast, the HAR model yields only minor improvements in forecast accuracy when additional variables are included. Third, by investigating the structure of the ML methods we uncover a nonlinear interaction between covariates. Fourth, we compute variable importance via accumulated local effect (ALE) plots. It shows a general agreement about the set of the most dominant predictors, but disagreement on their ranking. Fifth, we investigate how ML techniques respond to pure noise variables.

The paper is organized as follows. Section 2 sets the theoretical foundation and introduces the various models applied in the paper. Section 3 describes the high-frequency data used in the empirical analysis. Section 4 features an out-of-sample comparison and a series of robustness checks. We conclude in Section 5. In the Appendix, we present supplemental information, including an ML glossary. We also conduct a sensitivity analysis for the flash crash of 24 August 2015.

2 Methodology

2.1 The setting

We assume the log-price $(X_t)_{t\geq 0}$ is defined on the filtered probability space $\big(\Omega, (\mathcal{F}_t)_{t\geq 0}, \mathcal{F}, \mathbb{P}\big)$. Allowing the price to be determined in an arbitrage-free frictionless market implies X_t is a semimartingale process, see Delbaen and Schachermayer (1994). This means the log-price at time t can be represented by the following form:

$$X_t = X_0 + \int_0^t \mu_s ds + \int_0^t \sigma_s dW_s + \sum_{i=1}^{N_t} J_i, \quad t \geq 0, \tag{1}$$

where X_0 is $\mathcal{F}_0$-measurable, $\mu = (\mu_t)_{t\geq 0}$ denotes the drift, $\sigma = (\sigma)_{t\geq 0}$ is the stochastic volatility process, $W = (W_t)_{t\geq 0}$ is a standard Brownian motion, $N = (N_t)_{t\geq 0}$ is a counting process, which represents the total number of jumps in X, and $J = (J_i)_{i=1,\ldots,N_t}$ is a sequence of nonzero random variables, which denotes the corresponding jump sizes.

Our aim is to forecast the daily quadratic variance (QV), which is formally defined as $QV_t = \int_{t-1}^t \sigma_s^2 ds + \sum_{t-1\leq i\leq t} J_i^2$, for $t = 1, \ldots, T$, where t is the day and T is the total number of days in the sample. The QV process is not directly observable in practice since only discretely sampled

measurements of X are available. An estimator of QV is given by the realized variance (RV):

$$RV_t = \sum_{j=1}^{n} |\Delta_{t-1,j}^n X|^2, \tag{2}$$

where n is the number of intraday logarithmic returns, $\Delta_{t-1,j}^n X = X_{t-1+\frac{j}{n}} - X_{t-1+\frac{j-1}{n}}$. Then, as $n \to \infty$, $RV_t \xrightarrow{p} QV_t$.

2.2 The HAR model

The parsimonious structure of the HAR model of Corsi (2009) makes it a default choice in the literature. Therefore, the HAR model is the benchmark throughout this study, ensuring comparability with earlier work, even though there now is a large number of extensions available, such as HAR-J by Andersen, Bollerslev, and Diebold (2007), tree-structured HAR by Audrino and Corsi (2010), LHAR-CJ by Corsi and Renò (2012), SHAR by Patton and Sheppard (2015) and HAR-Q by Bollerslev, Patton, and Quaedvlieg (2016). In Section 4.5, however, we change the benchmark model to a log-HAR imposing a nonlinear structure on volatility.

The HAR model is defined as:

$$RV_t = \beta_0 + \beta_1 RV_{t-1} + \beta_2 RV_{t-1|t-5} + \beta_3 RV_{t-1|t-22} + u_t, \tag{3}$$

where $RV_{t-1|t-h} = \frac{1}{h}\sum_{i=1}^{h} RV_{t-i}$. The explanatory variables are the daily, weekly, and monthly lags of volatility capturing the long-term dependency structure.

To ensure comparability with the ML methods, we also adopt an extended HAR model, where exogenous variables are included. This is called the HAR-X:

$$RV_t = \beta_0 + \beta' Z_{t-1} + u_t, \tag{4}$$

where $\beta = (\beta_1, \ldots, \beta_J)'$ and J is the number of explanatory variables in the dataset $\mathcal{M}$ (e.g. $Z_{t-1} = (RV_{t-1}, RV_{t-1|t-5}, RV_{t-1|t-22})$ and $J = 3$ for the standard HAR).

In order to estimate the coefficient of the HAR and HAR-X model, we employ a least squares approach with objective function:

$$\hat{\beta} = \arg\min_\beta \mathcal{L}(\beta) = \arg\min_\beta \sum_{t \in \text{in-sample}} \big(RV_{t+1} - f(Z_t;\beta)\big)^2, \tag{5}$$

where $f(Z_t;\beta) = \beta' Z_t$ and in-sample is defined below.

2.3 Regularization

When increasing the number of predictors in a setting with a low signal-to-noise ratio, the linear model starts to fit noise rather than relevant information, also known as overfitting. A common way

to avoid overfitting and increase out-of-sample performance is to shrink the regression coefficients by imposing a penalty term.

A penalized loss function is given by:

$$\tilde{\mathcal{L}}(\beta;\theta) = \mathcal{L}(\beta) + \phi(\beta;\theta), \tag{6}$$

where $\phi(\beta;\theta)$ is a penalty term, and θ is a vector of hyperparameters, which are always determined in the validation set.[1]

In this paper, we estimate three of the most widely used penalization methods, i.e. ridge regression (RR) by Hoerl and Kennard (1970) and lasso by Tibshirani (1996), where the penalty term is also referred to as the ℓ_2 and ℓ_1 norm, and elastic net by Zou and Hastie (2005).

In ridge regression, the penalty term is given by:

$$\phi(\beta;\lambda) = \lambda \sum_{i=1}^{J} \beta_i^2, \tag{7}$$

where $\lambda \geq 0$ controls the amount of shrinkage.

If dealing with a large feature space, shrinking is not always sufficient, and subset selection is preferable. Lasso addresses this problem with a penalty term of the form:

$$\phi(\beta;\lambda) = \lambda \sum_{i=1}^{J} |\beta_i| \,. \tag{8}$$

In addition, due to the geometry lasso often forces many coefficient estimates exactly to zero and generates sparsity and subset selection. The cost of this advantage is that no closed-form solution exists, which makes it necessary to conduct numerical optimization.

The final penalization approach is the elastic net, which takes

$$\phi(\beta;\lambda,\alpha) = \lambda \left(\alpha \sum_{i=1}^{J} \beta_i^2 + (1-\alpha) \sum_{i=1}^{J} |\beta_i| \right), \tag{9}$$

where $\alpha \in [0,1]$.[2]

2.4 Tree-based regression

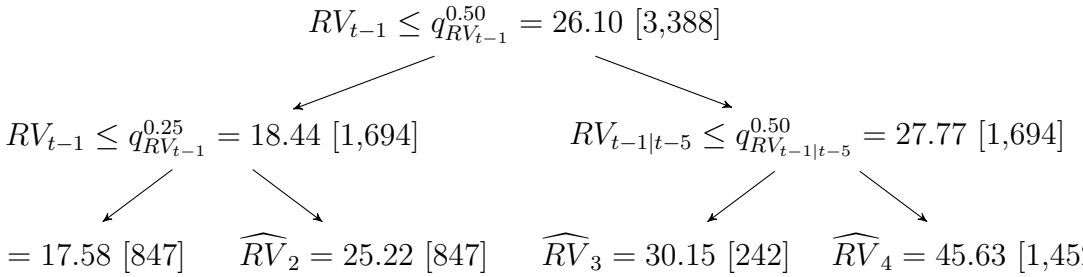
A concern with linear models is that it is left to the researcher to impose the true association among predictors and the response variable. In contrast, a regression tree is a fully nonparametric approach, which not only allows for nonlinearity, but it also implicitly accounts for interaction effects between explanatory variables.

A tree-based regression is, as the name suggests, based on a decision tree. A tree is grown by partitioning the domain (or feature space in ML parlance) of the explanatory variables Z_t,

[1] A thorough explanation of the validation procedure can be found in Section A.7 in the Appendix.

[2] The elastic net nests lasso for $\alpha = 0$ and ridge for $\alpha = 1$.

Figure 1: Illustration of a regression tree.



Note. The figure shows an example of a possible regression tree constructed with the actual values of RV_{t-1}, $RV_{t-1|t-5}$, and $RV_{t-1|t-22}$ from the training and validation set of Apple (AAPL) high-frequency data in our empirical application. The variables are expressed as annualized standard deviation in percent for ease of interpretation. We arbitrarily set a stopping criteria to a maximum number of terminal nodes $M = 4$ and select the split points conveniently as the 1st and 2nd quartile ($q^{0.25}$ and $q^{0.50}$) of the associated variable for the purpose of the illustration. The numbers in square brackets are the total data points retained at each node, as we move through the tree, while $\widehat{RV}_m$, $m = 1, \ldots, 4$, corresponds to the one-day-ahead prediction of RV_t at each terminal node. Note the uneven distribution of data in the right-hand side of the tree is caused by using the overall median of $RV_{t-1|t-5}$ as a split point.

$\text{dom}(Z_t) \subseteq \mathbb{R}^J$, into smaller and smaller rectangular subspaces as we move through the tree (an illustrative example is provided in Figure 1). This process is continued until a stopping criterion is reached. At the end of the tree, a constant prediction is assigned to all observations that fall in a given terminal node (also called a leaf).

Suppose that after constructing the tree there are M terminal nodes. They correspond to a sequence of disjoint rectangles R_m, for $m = 1, \ldots, M$, such that $R_m \subseteq \text{dom}(Z_t)$ and $\bigcup_{m=1}^{M} R_m = \text{dom}(Z_t)$. The tree-based regression then predicts as follows:

$$\hat{f}(Z_t) = \sum_{m=1}^{M} \widehat{RV}_m 1_{\{Z_t \in R_m\}}, \tag{10}$$

where $\widehat{RV}_m$ is a constant.

To ensure internal consistency, we adopt the minimum sum of squares criterion for selecting $\widehat{RV}_m$, i.e. $\widehat{RV}_m = \text{average}(RV_{t+1} \mid Z_t \in R_m)$.

In essence, tree-based regression poses a combinatorial problem. There is no closed-form solution to find the best order in which to sort and split variables, and it may be too time-consuming to search over all possible choices. Hence, the need for a short-sighted, greedy algorithm arises. A popular approach to this problem is to start at the top node (called the root) and independently optimize nearby nodes in a stepwise fashion, known as the Classification And Regression Tree (CART). The CART, however, often yields inferior out-of-sample performance due to high variance from overfitting sample noise.

Breiman (1996) proposed bagging (bootstrap aggregation) to refine the decision tree. In bagging, we grow several trees by resampling the original data. We subsequently predict as follows:

$$\hat{f}(Z_t) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(Z_t^b), \tag{11}$$

where $\hat{f}^b(Z_t^b)$ is the tree-based prediction from the bth bootstrap sample.

The serial correlation in volatility suggests a block bootstrap is appropriate in our context. Nonetheless, in our experience a standard i.i.d. bootstrap does not affect the recursive segmentation of the input space compared to more refined resampling.[3]

The benefit of bagging compared to CART is related to the reduced correlation between trees. Therefore, an alternative approach building on bagging is Random Forest (RF), see Breiman (2001). Here, attention is restricted to a random subset of input features before finding the best split point. In this way a random forest is able to de-correlate the trees further. In particular, B trees $\left\{f(Z_t^b, \theta^b)\right\}_{b=1}^B$ are grown, where θ^b summarizes the bth random forest tree in terms of split variables, split points, and values at the terminal nodes. The prediction is then:

$$\hat{f}(Z_t) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(Z_t^b, \theta^b). \tag{12}$$

To reduce the complexity of the random forest and bagging, all tuning parameters are set to the default values from the original Fortran code by Breiman and Cutler (2004), see Table 15 in the Appendix. This sidesteps validation and facilitates reproduction of our findings.

Another tree-based approach is Gradient Boosting (GB) introduced by Friedman (2001).[4] Gradient boosting produces a sequential model based on weak learners, where each of B trees is grown on information from the previous one. Firstly, $\hat{f}^0(Z_t)$ is initialized as a constant determined by $\hat{f}^0(Z_t) = \arg\min_\rho \sum_{t\in\text{in-sample}} \mathcal{L}(RV_t, \rho)$. With the mean squared error as a loss function, this prediction corresponds to the average. Secondly, the negative gradient of the loss function with respect to the prediction (equivalent to the residuals) is calculated. Then a shallow tree is fitted to the residuals, which yields a set of terminal nodes R_{jb}, $j = 1, \ldots, J_b$, where j denotes the leaf and b the tree. This is followed by choosing a gradient descent size as $\rho^{jb} = \arg\min_\rho \sum_{Z_t\in R_{jb}} \mathcal{L}(RV_t, \hat{f}^{b-1}(Z_t) + \rho)$, and in the last step $\hat{f}^b(Z_t)$ is updated iteratively as follows:

$$\hat{f}^b(Z_t) = \hat{f}^{b-1}(Z_t) + \nu \sum_{j=1}^{J_b} \rho^{jb} 1_{\{Z_t\in R_{jb}\}}, \tag{13}$$

for $b = 1, \ldots, B$, where ν is a learning rate and $\hat{f}(Z_t) \equiv \hat{f}^B(Z_t)$ the final prediction.

As explained by de Prado (2018), gradient boosting attempts to address an underfitting problem, whereas random forest deals with overfitting (generally perceived to be a greater evil in finance). As a consequence, gradient boosting places substantially more weight on misclassification and is therefore susceptible to outliers. With the typical heavy tails in the volatility distribution, we expect this to lead to poor results for gradient boosting in our framework.

[3] As a verification, we applied a block bootstrap with B random samples from a given dataset $\mathcal{M}$, resulting in a sequence $\mathcal{M}^b$, for $b = 1, \ldots, B$, where the block length l and the number of blocks in each bootstrap sample k was chosen such that $T = kl$. l was found in the validation set. We estimated a regression tree for each bootstrap and computed the prediction function as $\hat{f}(Z_t) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(Z_t)$. There was no discernible change in the results compared to the i.i.d. bootstrap.

[4] He extends the original boosting algorithm from Schapire (1990) and Freund (1995).

2.5 Neural network

To ensure a complete investigation of ML space, at last we apply the artificial neural network. With their highly flexible and nonlinear structure, neural networks show astonishing performance in handling complex problems. However, with a broad set of hyperparameters the replicability of the neural network is frequently questioned. Therefore, we construct the neural network as streamlined as possible and set the majority of the hyperparameters to standard suggestions from the literature. Our choices are presented in Table 15 in the Appendix.

We briefly outline the mechanics of a so-called feed-forward network, while a more comprehensive overview is available in, e.g., Goodfellow, Bengio, and Courville (2016). A neural network is constructed from an input layer, hidden layers, and an output layer. Suppose the total number of such layers is L. In the first layer, the neural network receives an input Z_t. The data is transformed through one or more hidden layers using an activation function g.

The general equation for the l-th layer is denoted as:

$$a_t^{\theta_{l+1},b_{l+1}} = g_l\left(\sum_{j=1}^{N_l} \theta_j^{(l)} a_t^{\theta_l,b_l} + b^{(l)}\right), \quad 1 \leq l \leq L, \tag{14}$$

where g_l is the activation function, $\theta^{(l)}$ is the weight matrix, $b^{(l)}$ is the error serving as an activation threshold for the neurons in the next layer, N_l denotes the number of hidden neurons, and $a_t^{\theta_{l+1},b_{l+1}}$ is the prediction.[5]

The output layer constructs a forecast:

$$\hat{f}\left(Z_t\right) = \left(g_L \circ \cdots \circ g_1\right)\left(Z_t\right). \tag{15}$$

As seen, a neural network applies a series of functional transformations to construct the forecast. In principle, a single hidden layer with a large enough number of neurons (and appropriate activation function) is sufficient to approximate any continuous function. This follows from the Universal Approximation Theorem of Cybenko (1989). In practice, however, it is often convenient and computationally more efficient to add extra hidden layers than to arbitrarily increase the number of neurons in a layer. Hence, the optimal architecture of a neural network depends on the problem and is determined through tuning.

To allow further inspection of the inner workings of a neural network, we construct four models with an architecture inspired by the geometric pyramid (see, e.g., Masters, 1993; Gu, Kelly, and Xiu, 2020). The first, denoted NN_1, has a shallow structure with a single hidden layer and 4 neurons. NN_2 is two-layered with 8 and 4 neurons, NN_3 has three layers with 16, 8, and 4 neurons, and NN_4 has four layers with 32, 16, 8, and 4 neurons.

We apply the Leaky Rectified Linear Unit (L-ReLU) by Maas, Hannun, and Ng (2013) as activation function for all layers, as it is able to infer when the activation is zero via gradient-based

[5]By definition, the prediction from the input layer is $a_t^{\theta_2,b_2} = g_1\Big(\sum_{j=1}^{N_1} \theta_j^{(1)} Z_t + b^{(1)}\Big)$.

methods, in contrast to the standard ReLU.[6] The L-ReLU is defined as:

$$\text{L-ReLU}(x) = \begin{cases} cx, & \text{if } x < 0, \\ x, & \text{otherwise,} \end{cases} \tag{16}$$

where $c \geq 0$.[7] The weight parameters are chosen to minimize the squared error loss function.[8] In lack of a closed-form solution, we adopt the Adaptive Moment Estimation (Adam) by Kingma and Ba (2014), where the optimizer updates the weights $(\hat{\theta}, \hat{b})$ as a combination of Momentum and RMSprop (Root Mean Square propagation).[9] We further predetermine the learning rate at 0.001, see Kingma and Ba (2014).

Regularization is crucial to account for overfitting in a neural network. There is a large literature proposing different regularization techniques to find the optimal weights. In this study, we employ four of the most commonly used, namely learning rate shrinkage, drop-out, early stopping, and ensembles. Section A.7 – A.8 in the Appendix explains in more detail the choice of hyperparameters and the various regularization techniques.

2.6 Forecast comparison

We contrast the out-of-sample performance of each method against the HAR model. As the mean squared error (MSE) laid the foundation for the optimal structure of the ML algorithms, a logical continuation is to employ it also as an out-of-sample evaluation measure:

$$\mathcal{L}(\hat{f}(Z_t)) = \sum_{t \in \text{out-of-sample}} \left(RV_{t+1} - \hat{f}(Z_t)\right)^2, \tag{17}$$

where $\hat{f}(Z_t)$ is the model dependent forecast.

To measure statistical significance, we construct a Model Confidence Set (MCS) of Hansen, Lunde, and Nason (2011). It defines a collection of models containing the "best" one with a given level of confidence. Inferior forecasting models are removed via an elimination rule.

3 Data description

The empirical investigation is based on high-frequency data from 29 Dow Jones Industrial Average (DJIA) constituents that traded continuously from 29 January 2001 to 31 December 2017, or 4,235

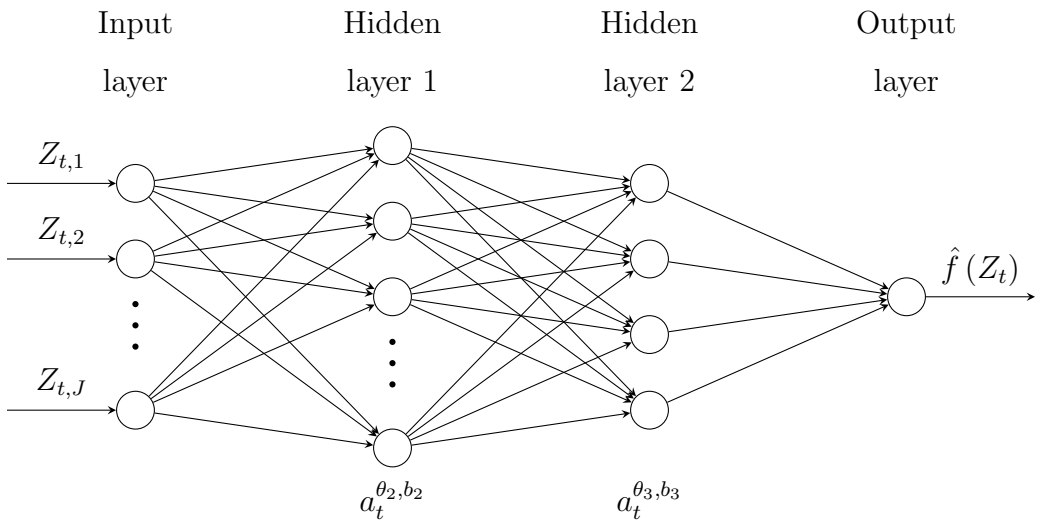
[6] Other common activation functions include Sigmoid, TanH, or ReLU. Switching to one of these has no material impact on our empirical results.

[7] In this paper, $x = \sum_{j=1}^{N_l} \theta_j^{(l)} a_t^{\theta_l, b_l} + b^{(l)}$. We follow the majority of the literature by setting $c = 0.01$.

[8] Alternatively, a penalized ℓ_2 objective function can be applied. We found no additional improvement when combining ℓ_2 and drop-out.

[9] To limit the number of hyperparameters and reduce the complexity of the optimization, we select exponential decay rates in line with Kingma and Ba (2014).

Figure 2: Feed-forward neural network.



Note. We illustrate a two-layered feed-forward neural network with an input layer, a pair of hidden layers, and the output layer. The input layer receives the raw features Z_t with J explanatory variables. Each hidden layer computes an affine transformation and applies an element-wise activation function. The output layer returns the prediction. The arrows denote the direction of the information flow through the network during forward propagation.

trading days. Visa (V) is excluded due to limited data availability.[10] The sample includes several noteworthy events, such as the financial crisis in 2008 – 2009, the European sovereign debt crisis, several rounds of dispute about the U.S. debt ceiling, and the flash crashes of 6 May 2010 and 24 August 2015. The data is extracted from the NYSE Trade and Quote (TAQ) database. We restrict attention to the official trading hours from 9:30 to 16:00 EST and compute for each asset a time series of daily realized variance RV_t, for $t = 1, \ldots, 4{,}235$, based on five-minute log-returns (i.e. $n = 78$). The goal is to forecast one-day-ahead realized variance.[11]

We split the data into a training, validation, and test set. The primary analysis is restricted to a training set of 70% of the dataset, a validation set of 10%, and a test set of 20%. The latter corresponds to a total of 847 days and is reserved for out-of-sample evaluation. We conduct a robustness check, where the test set is modified to 15% and 25% (keeping the validation set fixed at 10%). The results, which are broadly in line with our reported findings, are available in Section A.1 in the Appendix.

In the HAR and HAR-X model, we merge the training and validation set and employ time-varying parameters by constructing a rolling window forecast.[12] This approach is also adopted

[10]The shares of Visa were only listed on NYSE on March 19, 2008.

[11]In an unreported analysis, we attempted to forecast aggregate DJIA equity index volatility based on cross-sectional information of the volatilities of the individual stocks. However, we did not find any significant gain in forecast accuracy, once we controlled for past market-wide volatility.

[12]We also re-estimated the HAR on a daily basis from a rolling window of 1,000 observations, as in Corsi (2009).

for bagging and random forest to avoid any hyperparameter tuning, which results in off-the-shelf implementation of these methods. However, tuning is almost unavoidable for ridge, lasso, elastic net, and gradient boosting. In keeping with the off-the-shelf approach, a rolling scheme without concatenation of the training and validation set is conducted. An in-depth explanation of our selection (or tuning) of hyperparameters is available in Section A.7. Since the neural network is criticized for its computational budget, we employ a fixed scheme where the weights are found once in the validation step and not rolled forward in the out-of-sample window, see Section 2.5. This implementation is strictly advantageous to the competing models, which are allowed to adapt to new information. It cannot be ruled out that better performance for the neural networks can be attained with a rolling window, but we leave it for future research.

To study the effects of tagging on additional explanatory variables, we construct three datasets $\mathcal{M}_1$ – $\mathcal{M}_3$. $\mathcal{M}_1$ contains the daily, weekly, and monthly lag of realized variance (RVD, RVW, and RVM). This ensures one-to-one comparison between the HAR and ML algorithms. $\mathcal{M}_3$ extends $\mathcal{M}_1$ by including nine other key variables examined in previous research. We select four firm characteristics: Implied Volatility (IV),[13,14] an indicator for earnings announcements (EA), 1-week momentum (M1W), and dollar trading volume (DVOL).[15] We include five macroeconomic indicators: the CBOE volatility (VIX) index, the Hang Seng stock index squared log-return (HSI), the Aruoba, Diebold, and Scotti (2009) business conditions (ADS) index, the US 1-month T-bill rate (US1M) and the Economic Policy Uncertainty (EPU) index from Baker, Bloom, and Nicholas (2016).

With twelve variables in $\mathcal{M}_3$, concerns regarding overfitting and multicollinearity start surfacing, which is not expected to improve the results of the linear regression models. To overcome such criticism, we construct a third dataset, called $\mathcal{M}_2$, which is more favorable to the linear models. In contrast to $\mathcal{M}_1$ and $\mathcal{M}_3$, $\mathcal{M}_2$ fits a separate least squares regression for each possible combination of the feature space of $\mathcal{M}_3$, also known as subset selection. The single best model is then selected with a BIC criteria. It turns out to contain the daily and weekly lag of realized variance, implied volatility, 1-week momentum, and the earnings announcement indicator. Hence, a total of five variables are included in $\mathcal{M}_2$.[16]

A list with the full list of explanatory variables and their acronyms is presented in Table 1. It

With MSE loss, there was no material change in the results.

[13]The data was downloaded from the Ivy DB OptionMetrics database, which estimates the implied volatility for each traded option contract. We followed the data cleaning suggested by Lei, Wang, and Yan (2020) to deal with erroneous data points.

[14]To account for the correlation between implied volatility and realized variance, we also estimated an auxiliary model: $IV_t = \beta_0 + \beta_1 RV_t + u_t$ and employed the residuals from the fitted regression in place of implied volatility. The results were roughly unchanged.

[15]As dollar trading volume is trending up over time, we first difference the series to account for possible nonstationarity.

[16]We should remark that if the evaluation criterion is changed from BIC to cross-validation error, the best subset selection model includes the standard three HAR variables.

Table 1: List of explanatory variables.

No.	Acronym	Explanatory variable	Included in dataset:		
1	RVD	Daily lag of realized variance	1	2	3
2	RVW	Weekly lag of realized variance	1	2	3
3	RVM	Monthly lag of realized variance	1		3
4	IV	Implied volatility		2	3
5	EA	Earnings announcement*		2	3
6	EPU	Economic policy uncertainty index			3
7	VIX	CBOE volatility index			3
8	US1M	US 1-month T-bill rate			3
9	HSI	Squared Hang Seng index log-return			3
10	M1W	1-week momentum		2	3
11	DVOL	Dollar trading volume**			3
12	ADS	Business conditions index			3

Note. * A dummy variable set equal to one if the company makes a pre-scheduled earnings announcement on a given day, zero otherwise.
** We compute first differences of this variable to account for possible nonstationarity.

also shows which datasets the various variables are included in. We standardize the input data with the mean and variance in the training set to improve the numerical optimization procedure for ridge, lasso, elastic net and the neural network.

4 Empirical results

In this section, we first compare the MSE of the ML models relative to the baseline HAR forecast. Second, to investigate the significance of our results we construct the MCS. We also dissect forecast accuracy into low and high states of volatility. Third, in Section 4.2 – 4.3 we compute the ALE function to discover key drivers of volatility and look in more detail at the marginal relationship of the predictors with the response variable and interaction effects between explanatory variables. Fourth, a robustness check is done by adding a large amount of noise variables to the dataset. Fifth, a setting with log-volatility is inspected in Section 4.5.

Table 2: Out-of-sample relative MSE for linear and tree-based regression.

Dataset	#var	HAR-X	Ridge	Lasso	EN	BG	RF	GB
$\mathcal{M}_1$	3	1.000	1.000	1.003	0.999	1.141	1.020	1.055
$\mathcal{M}_2$	5	0.996	0.948	0.962	0.941	1.034	0.928	1.036
$\mathcal{M}_3$	12	0.979	0.915	0.942	0.910	0.977	0.906	0.983

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, ridge, lasso, elastic net (EN), bagging (BG), random forest (RF) and gradient boosting (GB). #var is the number of variables included in each dataset.

4.1 Out-of-sample evaluation

In Table 2 – 3, we report the ratio between the out-of-sample MSE of the HAR-X and ML techniques relative to the HAR model. Note that the $\mathcal{M}_1$ dataset relies on lagged realized variance, so here the HAR-X and HAR are identical. The results indirectly illustrate the complexity and low signal-to-noise ratio in the underlying data, as inferred from the MSE ratio of the HAR-X and HAR for $\mathcal{M}_2$ – $\mathcal{M}_3$. Indeed, subset selection based on BIC is unable to lower the MSE much, as gauged from Table 2, where we notice only a minuscule decrease in MSE of 0.4% moving from $\mathcal{M}_1$ to $\mathcal{M}_2$. While this is somewhat surprising, it supports our remarks in Section 2.3 regarding lack of regularization leaving HAR exposed to in-sample overfitting.[17]

As such, ridge regression is expected to excel compared to the HAR-X model when including additional exogenous variables. This is confirmed in Table 2. The ℓ_2 regularization enforced by ridge reduces the average MSE of 8.5% for $\mathcal{M}_3$ compared to the HAR. Interestingly, ridge is on par with HAR for $\mathcal{M}_1$. The intuition is that lagged realized variance is a highly significant predictor of volatility, so with only those three variables available, there is no impending need to penalize the model.

The effect of shrinkage versus subset selection can be uncovered by contrasting lasso with ridge. As indicated in Table 2, ridge is marginally lower than lasso, which nevertheless still beats the benchmark as more predictors are added. Interestingly, combining ridge and lasso via the elastic net is the best tool for regularization.[18] It ranks on top of the linear predictions with a pronounced 9.0% reduction in MSE over the HAR model for $\mathcal{M}_3$. This tendency persists when we vary the forecast period in the interval $[15\%, 25\%]$ of the dataset, as indicated in Table 8 and 10 in the Appendix.

In Table 2, we also observe that the dimension of the feature space is critical in achieving good performance for the regression trees. Bagging is worse than ridge and lasso, whereas random forest is the best-in-class among the tree-based approaches with an overall 9.4% reduction in MSE against

[17]Moreover, if the Flash Crash of 24 August 2015 is removed from the sample, the MSE of the HAR-X is inferior to the HAR, see Table 13 in Appendix A.6.

[18]The average value of the weight parameter α is about 0.165 in our sample.

Table 3: Out-of-sample relative MSE for neural network.

Dataset	#var	#ensemble	NN_1	NN_2	NN_3	NN_4
$\mathcal{M}_1$	3	1	1.161	0.972	0.955	0.964
		5	1.024	0.967	0.950	0.955
		10	1.000	0.967	0.951	0.955
$\mathcal{M}_2$	5	1	0.916	0.906	0.906	0.892
		5	0.909	0.894	0.892	0.885
		10	0.907	0.894	0.893	0.885
$\mathcal{M}_3$	12	1	0.905	0.897	0.899	0.890
		5	0.894	0.886	0.885	0.884
		10	0.894	0.882	0.882	0.883

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for four neural networks (NN). #var is the number of variables included in each dataset. #ensemble is the number of combined neural network predictors.

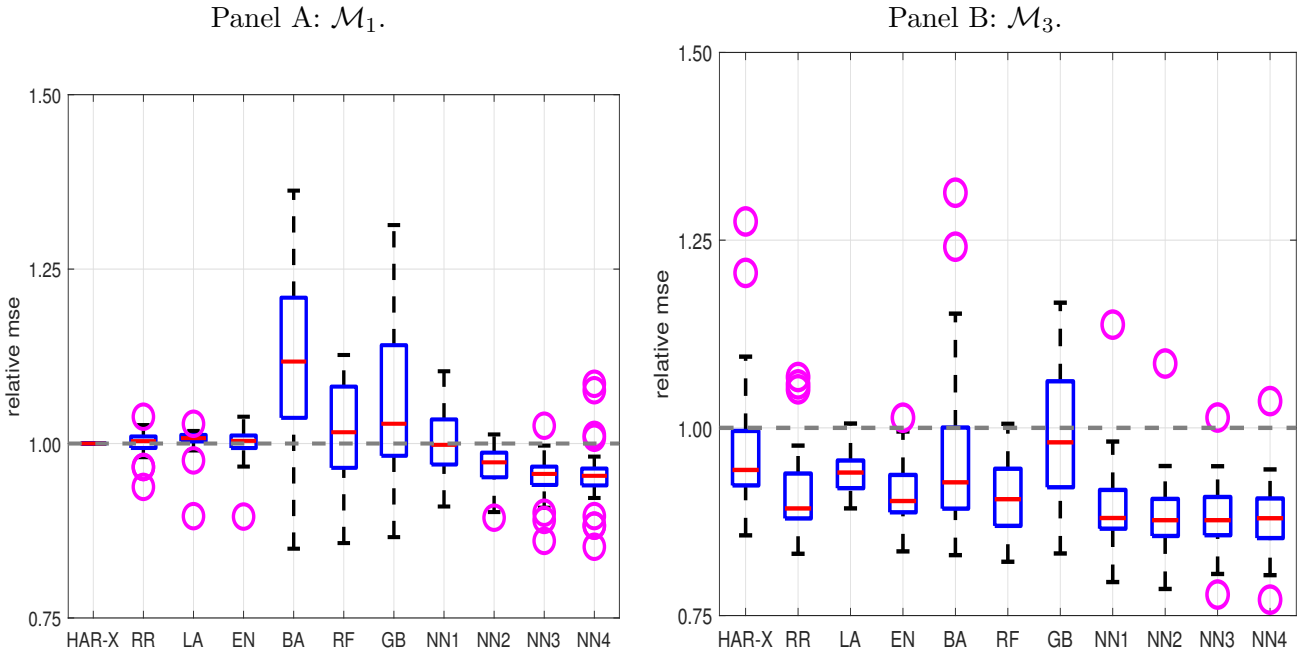
the HAR. This is arguably caused by its ability to decorrelate trees via random selection of the number of explanatory variables for the model. However, even though random forest has the lowest MSE among linear regression and tree-based predictors, there is only a marginal difference between it and the elastic net. This suggests that while allowing for automatic interactions between variables can be informative about future volatility, it also highlights that tackling the overfitting problem is the main cause of the sustained improvement.

If we compare random forest with gradient boosting, the latter shows inferior forecast accuracy. Gradient boosting produces a sequential model based on an ensemble of weak learners and, as stipulated above, places more weight on outlying observations in contrast to classical tree-based approaches. Hence, it is unsurprising that random forest does a better job than gradient boosting, given the low signal-noise ratio in the dataset.

Comparing the complex neural network to the benchmark HAR, it is expected that added flexibility increases forecast performance. As indicated in Table 3, it is evident that the average MSE for all neural networks decrease and that a crucial driver for this superior performance is the enlargement of the feature space. Remarkably, with the exception of the shallow NN_1, neural networks generally predict better than the HAR, even when only the lagged realized variance are included. This analysis is again supported by Table 9 and 11 in the Appendix, where the length of the test set is varied.

With an average MSE decrease of 11.5% in the $\mathcal{M}_3$-based prediction, the neural networks are best overall. The performance peaks with a reduction of 11.8% for NN_2 – NN_3 compared to the benchmark HAR. We should also point out that an additional decline in MSE is attainable by

Figure 3: Boxplot of cross-sectional out-of-sample relative MSE.



Note. We construct a boxplot for the one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, ridge regression (RR), lasso (LA), elastic net (EN), bagging (BA), random forest (RF), gradient boosting (GB), and four neural networks (NN). The sample consists of 29 relative MSEs for each model, corresponding to the number of stocks in our empirical application. The central mark is the median MSE, while the bottom and top edge of the box indicate the interquartile range (down: 25th percentile and up: 75th percentile MSE). The whiskers are the outermost observations not flagged as outliers. The latter are marked with a circle.

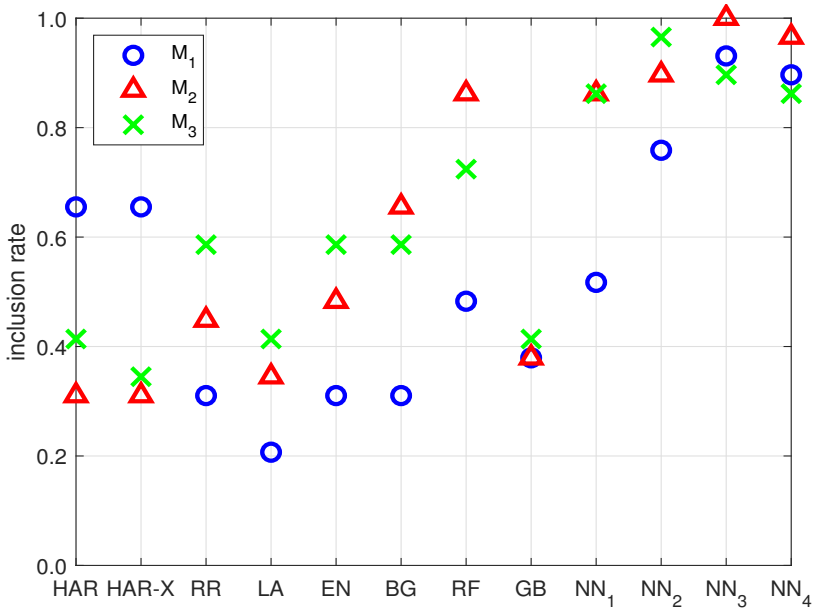
averaging the forecast over several different neural networks. This is called an ensemble, and we report the results of averaging 1, 5, and 10 neural networks.

To shed light on the causes of the decline in MSE for the neural networks, a logical next step is to inspect the importance of their complexity. In this aspect the geometric pyramid structure is helpful, as it means we can readily investigate the effect of changing the architecture of the network. Indeed, our results show that there is no big demand for an overly complex model, because the results are rather stable across network configurations.

In order to figure out whether the superior forecast accuracy of the ML methods is driven by a small subset of stocks or it extends to the entire cross-section, we dissect the aggregate numbers from Table 2 – 3. In Figure 3, we construct a boxplot of the relative MSE for each model versus the HAR across our sample of 29 stocks, both for dataset $\mathcal{M}_1$ in Panel A and $\mathcal{M}_3$ in Panel B (the results for $\mathcal{M}_2$ are roughly identical and not reported). As readily seen, there is a steady and sustained improvement in forecast accuracy of the ML methods as we grow the information set, even if the tree-based approaches are slightly inferior for $\mathcal{M}_1$.

In Figure 4, we compute the MCS at a 90% confidence level. The figure indicates the percentage of times each model stayed in the final set across stocks. It shows that random forest and neural networks generally rank on top with an inclusion rate of 72% and 90% for the $\mathcal{M}_3$ dataset, compared

Figure 4: Inclusion rate in the MCS.



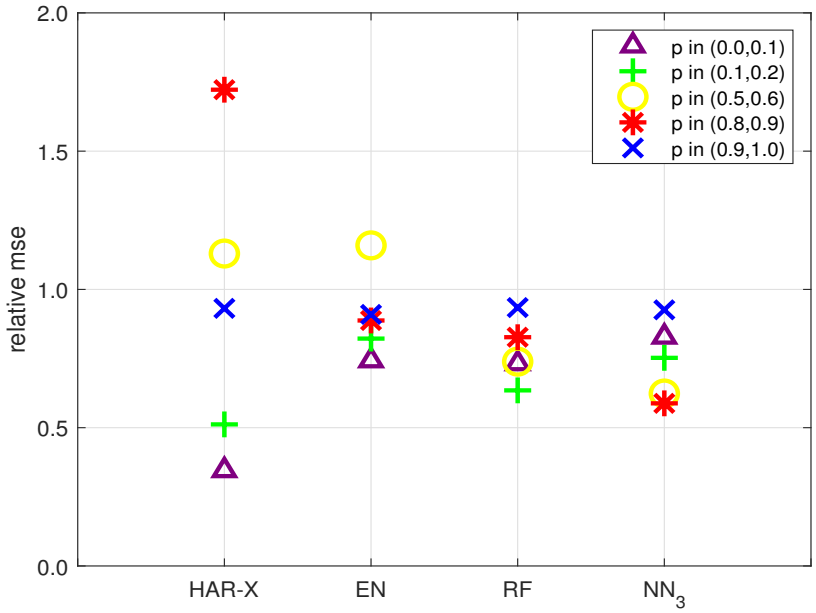
Note. We report the outcome of the Hansen, Lunde, and Nason (2011) MCS procedure, which identifies a subset of best models by running a sequence of tests for equal predictive ability, $H_0 : \text{MSE}_{\text{model}_i} = \text{MSE}_{\text{model}_j}$. The overall confidence level is set to 90% and the numbers indicate the percentage of times a given model is retained in the MCS.

to 41% for the HAR. Even as the feature space is expanded from three to twelve variables, the HAR-X still remains below 50%. Overall, the graph supports our previous finding regarding the superiority of the random forest and neural network.

To get insights into how the models forecast across various states of the volatility distribution, we split the test set into deciles according to the observed values of daily realized variance (i.e., the first subsample holds the 10% smallest values of RV_t in the out-of-sample window and is denoted (0.0,0.1)). For brevity we select the "preferred" model from each main category, i.e. we look at HAR-X, elastic net, random forest, and NN_3.

In Figure 5, we present the outcome of this effort for selected deciles. As indicated, the random forest and three-layered neural network yield lower MSE compared to the HAR both in times of high and low volatility. In particular, if we exclude the top decile of RV_t from the $\mathcal{M}_3$ dataset, we observe a remarkable reduction in the relative MSE of 36.04% for NN_3. Hence, the key driver for the excellent ML results can be found in the majority of the sample. The MSE is only reduced by 7.38% for the highest 10% of the RV_t sample, suggesting a closer alignment in forecast accuracy during market turmoil, which is of course crucial in practice. A possible explanation of this effect is offered in Section A.6 in the Appendix. Lastly, the sustained results for the random forest and neural network are not shared with the HAR-X and elastic net. Indeed, the HAR-X struggles when volatility is high, but has a much better MSE in calm markets. This indicates that when additional variables are present in the model, no regularization is required during low volatility.

Figure 5: Forecast accuracy over out-of-sample distribution of RV_t.



Note. The figure reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, elastic net, random forest, and NN_3. The sample is split into deciles of RV_t in the test set, where p stands for percentile. The dataset is $\mathcal{M}_3$.

4.2 Marginal effects and variable importance

The autonomy of a neural network or tree-based approach severely limits our ability to explain what is going on "under the hood", which often leads ML to be accused of being a black box approach. With their complex and highly nonlinear structure, a complete revelation of the underlying machinery is of course also rather difficult.

Nonetheless, we can illustrate the inner workings of the ML algorithms and shed at least some initial light on the black box by exploiting a recent contribution of Apley and Zhu (2020) that allows to compute an intuitive measure of marginal effect and variable importance in the ML framework. In particular, we calculate an Accumulated Local Effect (ALE) plot, which is an in-sample statistic based on the fitted model. To explain the concept, suppose for ease of exposition that $f(Z_t)$ is differentiable and that we are interested in analyzing the marginal effect of the jth predictor Z_{jt}, $j = 1, \ldots, J$, from the fitted model $\hat{f}(Z_t)$. We divide Z_t into Z_{jt} and the complement $Z_{\complement t} = Z_t \setminus Z_{jt}$, i.e. the remaining predictors in Z_t.

The ALE on $\hat{f}(Z_t)$ of Z_{jt} is the function:

$$\begin{aligned} f^{\text{ALE}}(z) &= \text{constant} + \int_{\min(z_j)}^{z} E\left[\frac{\partial \hat{f}(Z_{jt}, Z_{\complement t})}{\partial Z_{jt}} \mid Z_{jt} = x\right] \mathrm{d}x \\ &= \text{constant} + \int_{\min(z_j)}^{z} \int \frac{\partial \hat{f}(x, z_{\complement t})}{\partial x} p_{Z_{\complement t}|Z_{jt}}(z_{\complement t} \mid x) \mathrm{d}z_{\complement t} \mathrm{d}x, \end{aligned} \tag{18}$$

where $p_{Z_{\complement t}|Z_{jt}}(z_{\complement t} \mid x)$ is the conditional joint density of $Z_{\complement t}$ given $Z_{jt} = x$ and the function is defined over the range of observed values of Z_{jt} in the training set, as explained below. The constant is

such that $E[f^{\text{ALE}}(Z_{jt})] = 0$.

The term $\dfrac{\partial \hat{f}(x, z_{\complement t})}{\partial x}$ is called the local effect of Z_{jt} on $\hat{f}(Z_t)$. By accumulating the local effect, the value of ALE can be interpreted as the combined impact of Z_{jt} at the value z compared to the average prediction of the data. The derivative isolates the effect of the target predictor and controls for correlated features. Hence, ALE calculates changes in the prediction and adds them up, rather than computing the prediction directly. As explained in Apley and Zhu (2020) this circumvents problems with omitted variable biases. Moreover, by exploiting the conditional distribution the ALE is not impaired by unlikely data points, a key weakness of the more common Partial Dependence (PD) function of Friedman (2001).

f^{ALE} can be estimated via a finite difference approach. The information set used in the estimation is the matrix $Z = (Z_1, \ldots, Z_{T_0})$, where T_0 is the number of days in the training sample. The observations of the jth predictor correspond to the entries in the jth row of Z, which we denote $Z_j = (z_{j1}, \ldots, z_{jT_0})$. $Z_{\complement}$ is the complement of Z_j, i.e. the remaining rows in Z. We partition the range of Z_j such that $\min(Z_j) - \epsilon = z_0 < z_1 < \cdots < z_K = \max(Z_j)$, where ϵ is a small positive number ensuring that $\min(Z_j)$ is included in the subsequent calculations.[19] Next, as a function of $z \in (z_0, z_K]$ denote by $\text{idx}(z)$ the index of the interval into which z falls, i.e. $z \in (z_{\text{idx}(z)-1}, z_{\text{idx}(z)}]$. The uncentered ALE is estimated as follows:

$$\tilde{f}^{\text{ALE}}(z) = \sum_{k:z_k \leq z} \frac{1}{T_k} \sum_{t:z_{jt} \in (z_{k-1}, z_k]} \left[\hat{f}(z_k, z_{\complement t}) - \hat{f}(z_{k-1}, z_{\complement t}) \right], \tag{19}$$

where T_k is the number of data points of Z_j that falls in $(z_{k-1}, z_k]$ with $T_0 = \sum_{k=1}^{K} T_k$. As readily seen, the second summation approximates the local effect across small intervals by averaging over complement predictors $Z_{\complement}$ that are associated with the observations of $z_{jt} \in (z_{k-1}, z_k]$. The first sum accumulates these effects.

The centered ALE is then:

$$\hat{f}^{\text{ALE}}(z) = \tilde{f}^{\text{ALE}}(z) - T_0^{-1} \sum_{t=1}^{T_0} \tilde{f}^{\text{ALE}}(z_{jt}). \tag{20}$$

An ALE plot is based on the graph of $(z, \hat{f}^{\text{ALE}}(z))$.

To convert ALE to a measure of variable importance, we follow Greenwell, Boehmke, and McCarthy (2018) (in the PD framework) by computing the sample standard deviation of $\hat{f}^{\text{ALE}}$, i.e.

$$I(Z_j) = \sqrt{\frac{1}{T_0 - 1} \sum_{t=1}^{T_0} \left[\hat{f}^{\text{ALE}}(z_{jt}) \right]^2}. \tag{21}$$

Intuitively, if our prediction is only weakly related to Z_j, once we control for the effects of the other predictors, Z_j has a low importance. Consistent with this interpretation, the importance score is then $I(Z_j) \simeq 0$. On the other hand, if $\hat{f}^{\text{ALE}}$ fluctuates a lot over its domain, it means that Z_j has a large influence on the response variable, leading to larger values of $I(Z_j)$.

[19] In our implementation, we partition Z_j into 100 subintervals with equally many observations in each one.

Proceeding in this fashion, we construct a sequence $I(Z_j)$, for $j = 1, \ldots, J$. Our measure of variable importance is based on expressing each $I(Z_j)$ relative to their sum:

$$VI(Z_j) = \frac{I(Z_j)}{\sum_{j=1}^{J} I(Z_j)}, \quad \text{for } j = 1, \ldots, J. \tag{22}$$

Note that $VI(Z_j)$ ranges from zero to one.

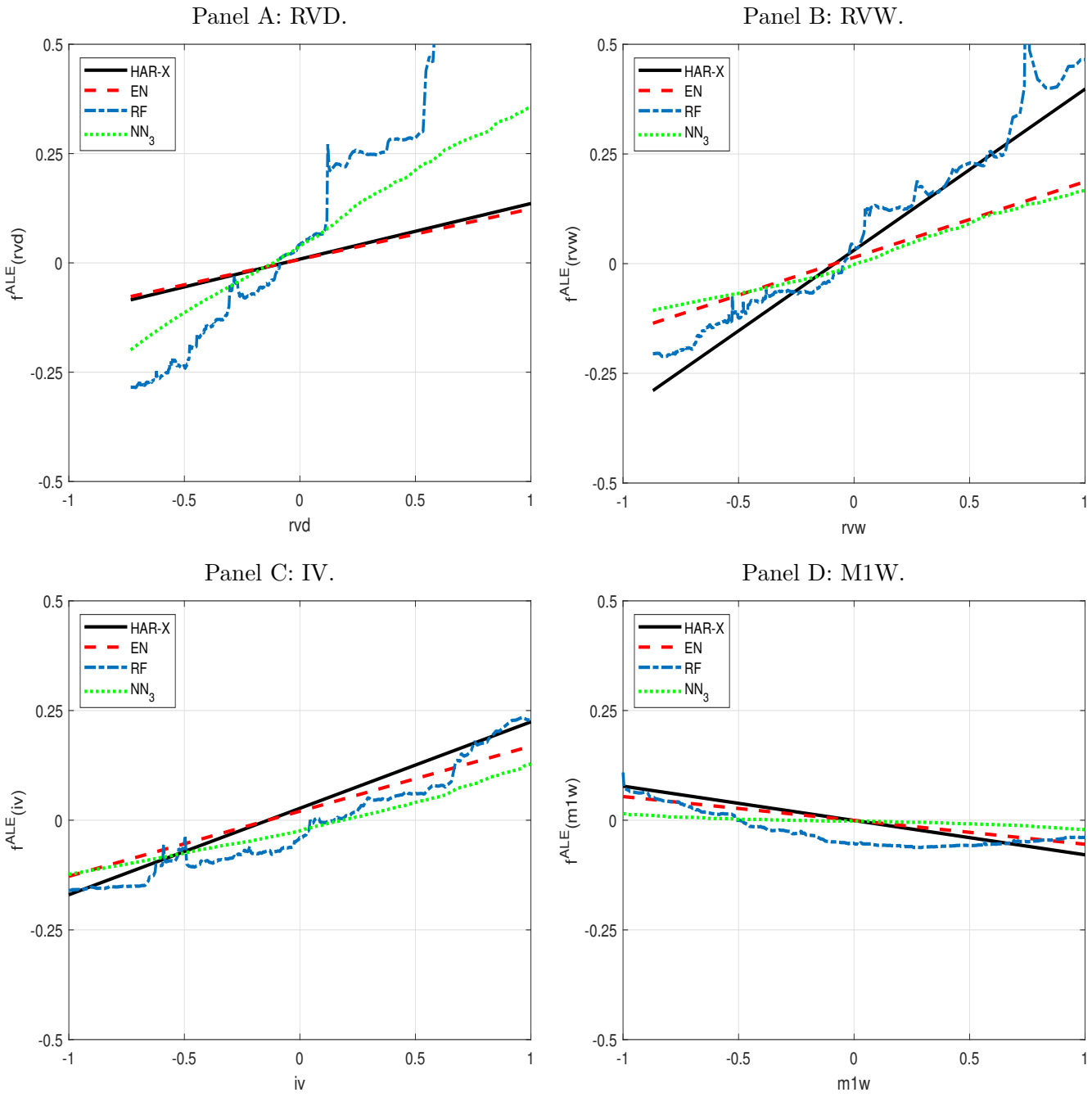
We do not explore the marginal effect averaged over stocks, as it produces unwarranted smoothing due to the unique relationship at the individual asset level. Instead, we arbitrarily select Apple for the illustration. Hence, this part serves as a showcase and may not be representative for the whole sample. In the construction of the ALE we restrict attention to a set of selected characteristics that are the most influential drivers of volatility, namely RVD, RVW, IV and M1W. Furthermore, we again select a single model from each category as above.

In Figure 6, we plot the ALE on expected realized variance of each variable over the interval [-1,1]. As expected, the figures reveal a positive relationship between RV_t and RVD, RVW, and IV. In contrast, the marginal association between RV_t and M1W is flat or even negative. This is supported by Table 12 in the Appendix, where parameter estimates for the HAR-X model of Apple are reported. Consistent with Figure 6, the slope coefficient for M1W is negative and significant. Secondly, the figures illustrate that allowing for a purely data-driven approach leads to a slightly nonlinear model. It also reveals that the marginal effect detected by the random forest and NN_3 are often very close, especially for RVD. Interestingly, the ML algorithms detect a stronger relationship between RV_t and RVD compared to the linear models, whereas the impact of RVW is attenuated. The discrepancy between elastic net and HAR-X for RVW is caused by the shrinkage of the parameter estimate of RVW, such that elastic net ends up suggesting a slope that is consistent with the neural network estimate.

The VI measure is reported for each variable in Figure 7. We concentrate on the HAR-X, elastic net, random forest, and NN_3, while the remaining models are presented in Section A.3 in the Appendix. As illustrated, across models there is a general agreement about the dominant set of predictors: RVD, RVW, and IV. This is intuitive, as these variates relate to the recent history or future expectation of the stock's volatility. In contrast, less weight is assigned to variables that are not directly related to volatility. However, even if there is consensus about the best set of predictors, there are individual differences in both their ranking and relative importance. The linear HAR-X model chooses RVW as the most important driver, whereas elastic net selects RVD, since it penalizes RVW. The nonlinear ML methods select both RVW and IV. For instance, NN_3 yields an ordering with RVW in the top spot, followed by IV, RVD and M1W. Hence, it weighs expectations of market participants higher than the daily and monthly lag of realized variance, which adds an ML perspective to the literature on informational content of realized variance versus implied volatility (e.g., Christensen and Prabhala, 1998; Busch, Christensen, and Nielsen, 2011).

With a nonlinear market structure, valuable information can potentially be extracted from a broader set of covariates. In line with this, we see that for instance ADS and US1M contribute more to the prediction of future volatility in the random forest relative to RVM, whereas these variables

Figure 6: ALE between explanatory variable and future volatility.



Note. This figure plots the Accumulated Local Effect (ALE) between RV_t and a single explanatory variable, i.e. RVD, RVW, IV, and M1W for Apple's stock price volatility. Note that the covariates are standardized. If the function does not cover the entire domain of the x-axis, it is because there are no observations in this area. The y-axis is kept constant to illuminate the relative importance of each variable in the prediction.

do not weigh much in the linear HAR-X. This is consistent with the results for the parameter estimates of the HAR-X model in Table 12 in the Appendix, where the coefficients in front of ADS and US1M are found to be insignificant in the $\mathcal{M}_3$ dataset.

As gauged in the Appendix, lasso applies subset selection. Accordingly, a sparse solution with seven beta coefficients restricted to zero is computed. Thus, lasso is capable of tackling overfitting, but it cannot extract incremental information from the majority of the explanatory variables as opposed to elastic net, random forest or neural network.

The EA variable has a low VI score. This highlights a weakness with the ALE analysis, since it is mainly a tool for finding the drivers behind the average prediction. Earnings information is released infrequently, but it is arguably very important for making short-run predictions of realized variance in the days surrounding the announcement.

In Section A.4 in the Appendix, we propose alternative measures of variable importance. The first is an in-sample measure, which calculates the change in the adjusted R^2 by dropping each variable one at a time (setting the values of predictor j equal to zero while holding the remaining model estimates fixed), whereas the second calculates the associated relative increase in out-of-sample MSE. The conclusions from those measures are broadly in line with those presented here.

4.3 Interaction effects

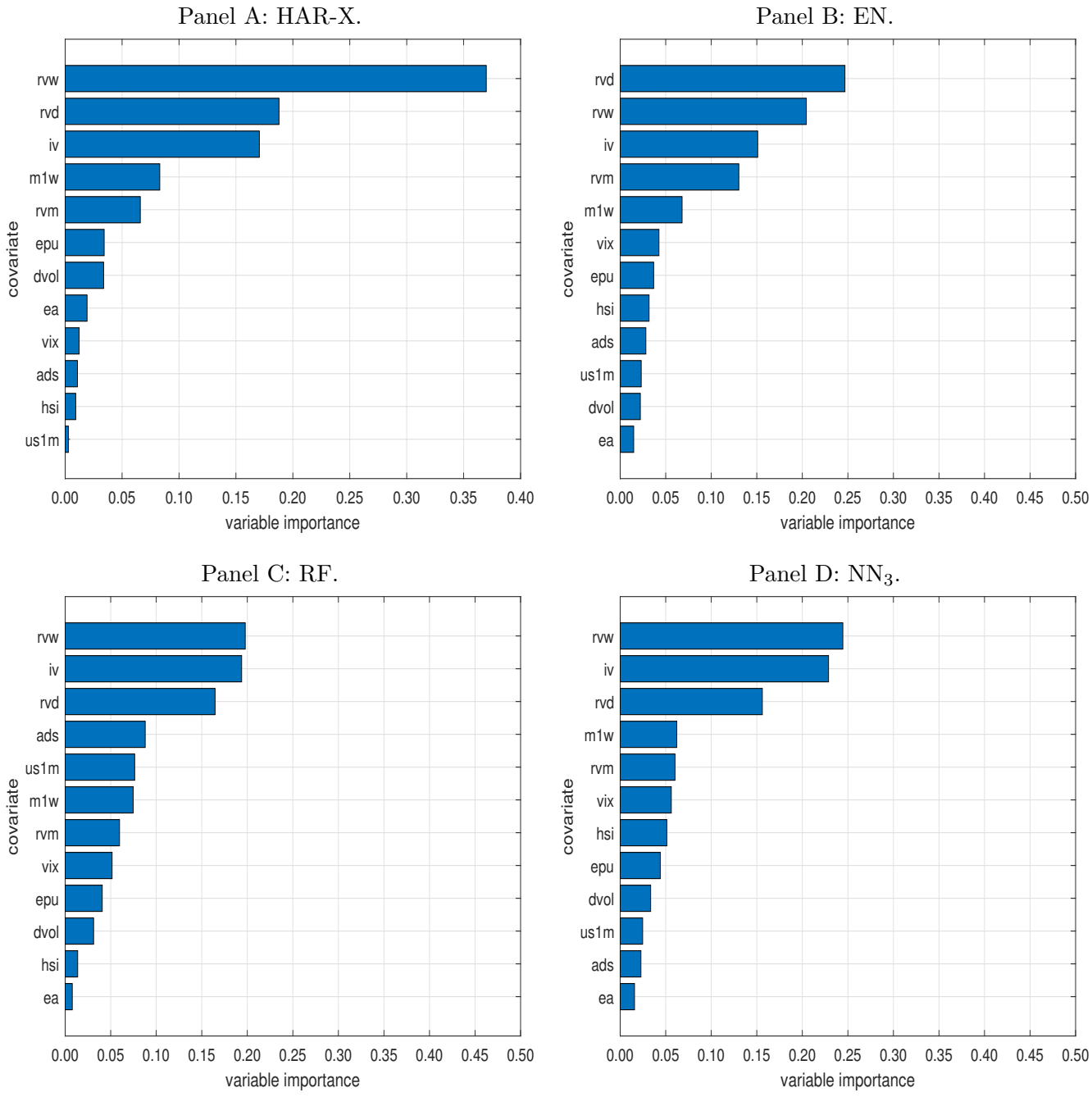
On top of allowing the conditional expectation of the response variable to be a nonlinear function of the covariates, neural networks automatically add nonlinear interaction effects between the explanatory variables to the model. Hence, Figure 6 is an incomplete picture of the underlying data. As a supplement, we construct Figure 8 that reports how expected realized variance changes when we vary two variables, holding the others fixed at zero. To narrow down the analysis, we only investigate the interaction between IV against one of RVD, RVW, and VIX in Panel A – C. In addition, we report the interaction between RVD and RVW in Panel D of the figure. In line with Section 4.2, we again look at Apple. Several interesting findings emerge. For example, from Panel A in Figure 8 we note that as IV goes to one standard deviation above its mean, holding RVD fixed at 0.5, a substantial uptick in the forecast of realized variance is detected. The interpretation is that if investors are expecting changes in volatility (as gauged via the forward-looking IV measure) when the previous realized variance is only modestly high, the effect of past volatility is overruled, in contrast to a less pronounced impact when volatility is low. This highlights the existence of nonlinear interaction effects.

4.4 Garbage in, garbage out?

As described above, one of the pitfalls of applying ML in financial markets is the low signal-to-noise ratio. This leads to concerns about their behavior in a high-dimensional setting of many variables with no informational content.

To assess the resilience of our models, we repeat our analysis with three datasets that extend $\mathcal{M}_3$ by adding varying degrees of pure noise. The new datasets are called $\mathcal{M}_4$ – $\mathcal{M}_6$ and include an additional 9, 45, and 90 noise terms. The construction of the variables is explained in Section

Figure 7: VI measure.

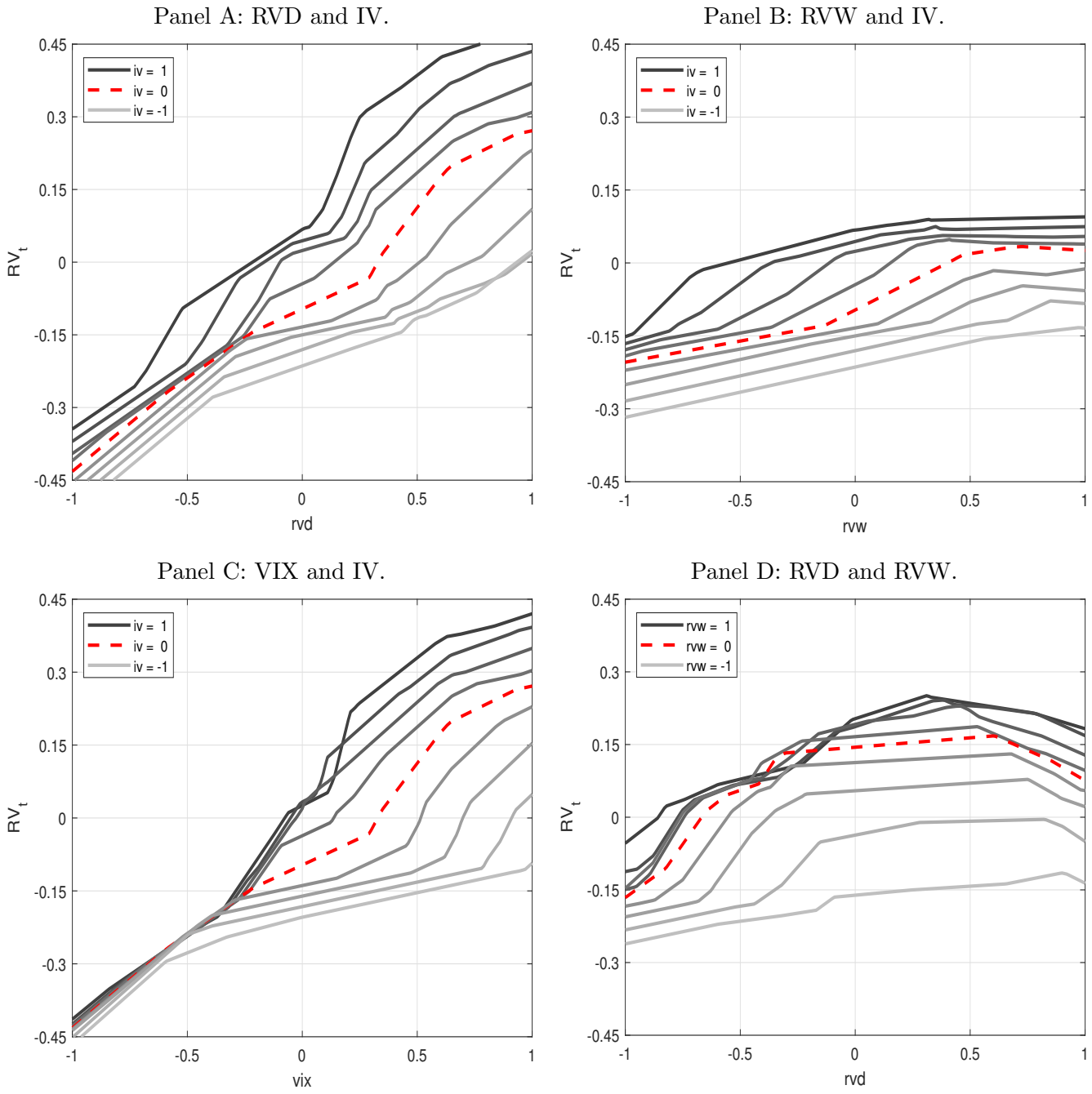


Note. We report the VI measure for each feature of HAR-X, elastic net, random forest and NN_3, sorted according to the numeric value of VI. The figures are based on high-frequency data from Apple.

A.2 in the Appendix. They are simulated independently of realized variance, but are scaled to a comparable level of volatility-of-volatility. Moreover, they are allowed to be both serially correlated and conditionally heteroscedastic, thereby emulating a clustering-type effect both in the mean and variance of the processes.

The results of the updated calculations are presented in Table 4 – 5. Surprisingly, there is

Figure 8: Interaction effects in the neural network.



Note. We plot the marginal effect on RV_t of pairwise interaction between explanatory variables in the NN_3 model. Each pair of covariates is varied in (-1,1): The first on the x-axis, the second by level of greying in the regression function. Other covariates are fixed at zero.

almost no change in the performance of the lasso and only a small increase of 0.09% percentage points for elastic net (comparing $\mathcal{M}_3$ to $\mathcal{M}_6$, which is the setting with the most dominant noise proportion). As expected, this suggests such procedures are very effective at removing noise. On the other hand, we detect a severe deterioration in MSE for other models when the number of noise terms is elevated. For instance, compared to the original information set HAR-X suffers heavily

Table 4: Out-of-sample relative MSE for linear and tree-based regression.

Dataset	#var	HAR-X	Ridge	Lasso	EN	BG	RF	GB
$\mathcal{M}_4$	21	1.003	0.927	0.941	0.914	0.951	0.911	1.047
$\mathcal{M}_5$	57	1.031	0.941	0.941	0.917	0.948	0.911	1.012
$\mathcal{M}_6$	102	1.078	0.964	0.941	0.919	0.977	0.914	1.046

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, ridge, lasso, elastic net (EN), bagging (BG), random forest (RF) and gradient boosting (GB). #var is the number of variables included in each dataset.

Table 5: Out-of-sample relative MSE for neural network.

Dataset	#var	#ensemble	NN_1	NN_2	NN_3	NN_4
$\mathcal{M}_4$	21	1	0.919	0.912	0.919	0.926
		5	0.901	0.896	0.900	0.907
		10	0.901	0.894	0.898	0.908
$\mathcal{M}_5$	57	1	0.949	0.946	0.939	0.947
		5	0.930	0.917	0.916	0.936
		10	0.930	0.911	0.917	0.939
$\mathcal{M}_6$	102	1	1.010	1.020	0.951	0.966
		5	0.949	0.962	0.935	0.949
		10	0.950	0.954	0.934	0.946

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for four neural networks (NN). #var is the number of variables included in each dataset. #ensemble is the number of combined neural network predictors.

with an average increase in MSE of 9.9%.

Table 5 reports moderate increases in MSE for the neural networks. The MSE of NN_3 increases by around 5.2% for $\mathcal{M}_6$. Still, the neural network achieving the worst relative MSE is nevertheless better than the baseline HAR. The latter operates with only three informative explanatory variables: daily, weekly, and monthly lag of realized variance and is therefore not inflicted with noise. Thus, a better—and arguably also more realistic—comparison is based on the HAR-X, which is totally outclassed by the previously top-ranked NN_3 with a 14.4%-point margin. This stresses the need for alternative approaches to linear regression, such as ML, when forecasting volatility in a data-rich environment.

Overall, the results are compelling. ML is able to decipher a highly complex dataset and extract valuable information about volatility, even when looking for the signal in a noisestack.

Table 6: Out-of-sample relative MSE for log-linear and log-tree-based regression.

Dataset	#var	L-HAR-X	L-Ridge	L-Lasso	L-EN	L-BG	L-RF	L-GB
$\mathcal{M}_1$	3	1.000	0.997	0.996	0.994	1.114	1.105	1.026
$\mathcal{M}_2$	5	0.933	0.920	0.934	0.920	0.975	1.027	0.970
$\mathcal{M}_3$	12	0.897	0.892	0.898	0.894	0.957	0.983	0.942

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the L-HAR model for L-HAR-X, L-ridge, L-lasso, L-elastic net (L-EN), L-bagging (BG), L-random forest (L-RF) and L-gradient boosting (L-GB). #var is the number of variables included in each dataset. The prefix "L-" denotes that the model is estimated from log-volatility.

4.5 The L-HAR model

As emphasized above, the danger of overfitting and lack of accounting for nonlinear interactions between the explanatory variables and the response variable entail poor outcomes for the HAR and HAR-X. In contrast, the log-HAR is known to be smoother and implicitly imposes a nonlinear relationship between present and future volatility.[20] In this section, we re-run the whole analysis based on log-realized variance.[21] We denote the log-version of a volatility model with a pre-fix "L-", i.e. the log-HAR is L-HAR and so forth.

We compute a reduction in relative MSE of 0.3% when comparing forecasts from the standard HAR and L-HAR model.[22] Moreover, as seen in Table 6 – 7 applying L-HAR-X yields even further decreases in MSE when additional variables are added to the dataset, in line with the HAR-X. In the largest information setting, the L-HAR-X, L-EN, and L-NN_3 models produce virtually identical results.

To further enhance the analysis, Figure 9 computes the relative MSE of the L-HAR-X and L-NN_3, where the test set has been separated into decile splits according to the observed value RV_t. It shows that there is a common tendency for both models to produce worse forecasts, when volatility is in the tails of its distribution. The L-HAR-X outperforms the L-NN_3 during low volatility, but the opposite is true and L-NN_3 is preferable with medium-to-high variance. The two models are

[20]The log-version of the HAR model was also proposed by Corsi (2009).

[21]To ensure our results are robust against a broader suite of more recent HAR-type models, we also fitted the HAR-Q by Bollerslev, Patton, and Quaedvlieg (2016). The results were broadly in line with those reported in the text. In particular, we found a reduction of over 10% when comparing the relative MSE for the random forest and neural networks to the HAR-Q model.

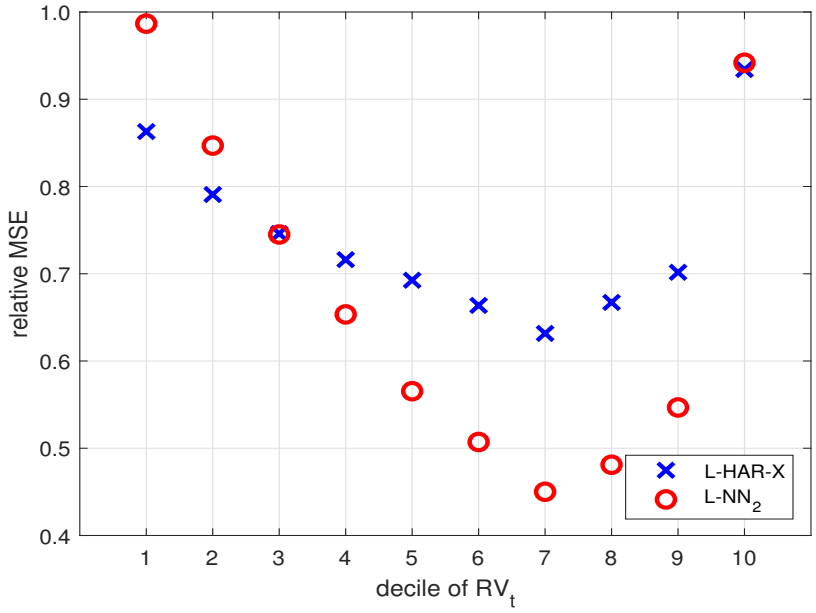
[22]In the log-setting we produce a forecast $\hat{f}(Z_t)$ of log-volatility, but the volatility is required $\exp\big(\hat{f}(Z_t)\big)$. We correct the log-volatility forecast via Jensen's inequality: $E\left[\exp\big(\hat{f}(Z_t)\big)\right] = \exp\Big(E\big[\hat{f}(Z_t)\big] + 0.5\text{var}\big[\hat{f}(Z_t)\big]\Big)$, which is suitable for Gaussian data. The distribution of realized variance is reportedly close to log-normal, see, e.g., Andersen, Bollerslev, Diebold, and Ebens (2001); Christensen, Thyrsgaard, and Veliyev (2019). $\text{var}\big(\hat{f}(Z_t)\big)$ is estimated from the unconditional variance of the residuals in the training and validation set.

Table 7: Out-of-sample relative MSE for log-neural network.

Dataset	#var	#ensemble	L-NN$_1$	L-NN$_2$	L-NN$_3$	L-NN$_4$
$\mathcal{M}_1$	3	1	0.990	0.980	0.976	0.978
		5	0.969	0.981	0.975	0.975
		10	0.968	0.982	0.976	0.976
$\mathcal{M}_2$	5	1	0.904	0.904	0.899	0.907
		5	0.900	0.904	0.898	0.902
		10	0.901	0.905	0.899	0.901
$\mathcal{M}_3$	12	1	0.898	0.882	0.881	0.885
		5	0.882	0.881	0.879	0.882
		10	0.880	0.882	0.879	0.883

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for four neural networks (NN). #var is the number of variables included in each dataset. #ensemble is the number of combined neural network predictors. The prefix "L-" denotes that the model is estimated from log-volatility.

Figure 9: Forecast accuracy over out-of-sample distribution of RV_t.



Note. The figure reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for L-HAR-X and L-NN$_3$. The sample is ordered based on the value of RV_t in the test set and split into deciles 1 – 10 (1: low, 10: high). The dataset is $\mathcal{M}_3$.

tied in the top decile. This supports our findings and shows that regularization and nonlinearities

are not required in a low volatility environment, but they are crucial when volatility shoots up.[23]

5 Conclusion

This paper synthesizes the field of machine learning (ML) with one-day-ahead volatility forecasting. We conduct an extensive out-of-sample forecast comparison of current ML methods with the HAR model from the realized variance literature. We show that ML can handle the often highly nonlinear structure governing financial markets and extract valuable information about future volatility in a data-rich environment with increasing feature space, which linear regression models struggle to do. The random forest and neural networks are found to be preferred among a broad range of ML algorithms and significantly improve prediction relative to HAR. As a robustness check, we add a substantial amount of noise variables to the information set in order to obfuscate the signal. We note the majority of the ML methods are fairly resistant to this effect, in contrast to the linear regression. We dissect the underlying structure of the data via ALE plots in order to pin down the main covariates leading to the improved prediction.

To the best of our knowledge, this study is the first to extensively implement a broad selection of the most widely applied ML methods and compare them for the problem of volatility prediction. We envision the paper can serve as a starting point for applying ML for forecasting stochastic volatility and perhaps even as a benchmark when building more complicated models.

[23]It deserves repetition that we do not tune hyperparameters in the neural network. In contrast to the L-HAR, we also do not allow for time-varying parameters by rolling the sample window and re-estimating the model. If permitted, we should expect an improvement in forecast accuracy of L-NN_3 not realized in the current setup.

References

Andersen, T. G., T. Bollerslev, and F. X. Diebold, 2007, "Roughing it up: Including jump components in the measurement, modeling and forecasting of return volatility," *Review of Economics and Statistics*, 89(4), 701–720.

Andersen, T. G., T. Bollerslev, F. X. Diebold, and H. Ebens, 2001, "The distribution of realized stock return volatility," *Journal of Financial Economics*, 61(1), 43–76.

Apley, D. W., and J. Zhu, 2020, "Visualizing the effects of predictor variables in black box supervised learning models," *Journal of the Royal Statistical Society: Series B*, 82(4), 1059–1086.

Aruoba, S. B., F. X. Diebold, and C. Scotti, 2009, "Real-time measurement of business conditions," *Journal of Business and Economic Statistics*, 27(4), 417–427.

Audrino, F., and F. Corsi, 2010, "Modeling tick-by-tick realized correlations," *Computational Statistics and Data Analysis*, 54(11), 2372–2382.

Audrino, F., and S. D. Knaus, 2016, "Lassoing the HAR model: A model selection perspective on realized volatility dynamics," *Econometric Reviews*, 35(8-10), 1485–1521.

Audrino, F., F. Sigrist, and D. Ballinari, 2020, "The impact of sentiment and attention measures on stock market volatility," *International Journal of Forecasting*, 36(2), 334–357.

Baker, S. R., N. Bloom, and S. J. Nicholas, 2016, "Measuring economic policy uncertainty," *Quarterly Journal of Economics*, 131(4), 1593–1636.

Bollerslev, T., 1986, "Generalized autoregressive conditional heteroscedasticity," *Journal of Econometrics*, 31(3), 307–327.

Bollerslev, T., J. Li, and Y. Xue, 2018, "Volume, volatility, and public news announcements," *Review of Economic Studies*, 85(4), 2005–2041.

Bollerslev, T., A. J. Patton, and R. Quaedvlieg, 2016, "Exploiting the errors: A simple approach for improved volatility forecasting," *Journal of Econometrics*, 192(1), 1–18.

Breiman, L., 1996, "Bagging predictors," *Machine Learning*, 24(2), 123–140.

——— , 2001, "Random forests," *Machine Learning*, 45(1), 5–32.

Breiman, L., and A. Cutler, 2004, "Random Forests," Fortran code version 5.1, available for download at `https://www.stat.berkeley.edu/ breiman/RandomForests/cc_software.htm`.

Bucci, A., 2020, "Realized volatility forecasting with neural networks," *Journal of Financial Econometrics*, 18(3), 502–531.

Busch, T., B. J. Christensen, and M. Ø. Nielsen, 2011, "The role of implied volatility in forecasting future realized volatility and jumps in foreign exchange, stock, and bond markets," *Journal of Econometrics*, 160(1), 48–57.

Caporin, M., and F. Poli, 2017, "Building news measures from textual data and an application to volatility forecasting," *Econometrics*, 5(3), 1–46.

Carr, P., L. Wu, and Z. Zhang, 2019, "Using machine learning to predict realized variance," arXiv preprint arXiv:1909.10035.

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

Christensen, B. J., and N. R. Prabhala, 1998, "The relation between implied and realized volatility," *Journal of Financial Economics*, 50(2), 125–150.

Christensen, K., M. Thyrsgaard, and B. Veliyev, 2019, "The realized empirical distribution function of stochastic variance with application to goodness-of-fit testing," *Journal of Econometrics*, 212(2), 556–583.

Corsi, F., 2009, "A simple approximate long-memory model of realized volatility," *Journal of Financial Econometrics*, 7(2), 174–196.

Corsi, F., and R. Renò, 2012, "Discrete-time volatility forecasting with persistent leverage effect and the link with continuous-time volatility modeling," *Journal of Business and Economic Statistics*, 30(3), 368–380.

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

de Prado, M. L., 2018, *Advances in Financial Machine Learning*. John Wiley & Sons, 1st edn.

Delbaen, F., and W. Schachermayer, 1994, "A general version of the fundamental theorem of asset pricing," *Mathematische Annalen*, 300(1), 463–520.

Donaldson, R. G., and M. Kamstra, 1997, "An artificial neural network-GARCH model for international stock return volatility," *Journal of Empirical Finance*, 4(1), 17–46.

Engle, R. F., 1982, "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation," *Econometrica*, 50(4), 987–1007.

Engle, R. F., E. Ghysels, and B. Sohn, 2013, "Stock market volatility and macroeconomic fundamentals," *Review of Economics and Statistics*, 95(3), 776–797.

Fernandes, M., M. C. Medeiros, and M. Scharth, 2014, "Modeling and predicting the CBOE market volatility index," *Journal of Banking and Finance*, 40(1), 1–10.

Freund, Y., 1995, "Boosting a weak learning algorithm by majority," *Information and Computation*, 121(2), 256–285.

Freund, Y., and R. E. Shapire, 1995, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, 55(1), 119–139.

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

Goodfellow, I., Y. Bengio, and A. Courville, 2016, *Deep Learning*. MIT Press, 1st edn.

Greenwell, B. M., B. C. Boehmke, J. P. Cunningham, and G. Developers, 2019, "Package "gbm"," Available at https://cran.r-project.org/web/packages/gbm/gbm.pdf.

Greenwell, B. M., B. C. Boehmke, and A. J. McCarthy, 2018, "A simple and effective model-based variable importance measure," Working paper.

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

Hansen, P. R., and A. Lunde, 2005, "A forecast comparison of volatility models: Does anything beat a GARCH(1,1)?," *Journal of Applied Econometrics*, 20(7), 873–889.

Hansen, P. R., A. Lunde, and J. M. Nason, 2011, "The model confidence set," *Econometrica*, 79(2), 453–497.

Hillebrand, E., and M. C. Medeiros, 2010, "The benefits of bagging for forecast models of realized volatility," *Econometric Reviews*, 29(5-6), 571–593.

Hoerl, A. E., and R. W. Kennard, 1970, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, 12(1), 55–67.

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

Lei, Q., X. W. Wang, and Z. Yan, 2020, "Volatility spread and stock market response to earnings announcements," *Journal of Banking and Finance*, 119(1), Article 105126.

Luong, C., and N. Dokuchaev, 2018, "Forecasting of realised volatility with the random forests algorithm," *Journal of Risk and Financial Management*, 11(4), 1–61.

Maas, A. L., A. Y. Hannun, and A. Y. Ng, 2013, "," in *Proceedings of the 30th International Conference on Machine Learning*.

Masters, T., 1993, *Practical Neural Network Recipes in C++*. Academic Press, 1st edn.

Mittnik, S., N. Robinzonov, and M. Spindler, 2015, "Stock market volatility: Identifying major drivers and the nature of their impact," *Journal of Banking and Finance*, 58(1), 1–14.

Patton, A. J., and K. Sheppard, 2015, "Good volatility, bad volatility: Signed jumps and the persistence of volatility," *Review of Economics and Statistics*, 97(3), 683–697.

Rahimikia, E., and S.-H. Poon, 2020, "Machine learning for realised volatility forecasting," Working paper.

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

Schwert, G. W., 1989, "Why does stock market volatility change over time," *Journal of Finance*, 44(5), 1115–1153.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 15(1), 1929–1958.

Taylor, S. J., 1982, "Financial returns modeled by the product of two stochastic processes - a study of the daily sugar prices 1961-75," in *Time Series Analysis: Theory and Practice*, ed. by O. D. Anderson. North-Holland, Amsterdam, pp. 203–226.

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

Varian, H. R., 2014, "Big data: New tricks for econometrics," *Journal of Economic Perspectives*, 28(2), 3–28.

Zou, H., and T. Hastie, 2005, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320.

A Appendix

A.1 Span of test set

This section reports the out-of-sample relative MSE of each volatility model when the length of the test set is modified from 20% to 15% (Table 8 and 9) and 25% (Table 10 and 11).

20% to 15%:

Table 8: Out-of-sample relative MSE for linear and tree-based regression.

Dataset	#var	HAR-X	Ridge	Lasso	EN	BG	RF	GB
$\mathcal{M}_1$	3	1.000	0.998	1.002	0.999	1.140	1.018	1.062
$\mathcal{M}_2$	5	0.992	0.949	0.964	0.945	1.020	0.932	0.993
$\mathcal{M}_3$	12	0.975	0.917	0.947	0.917	0.963	0.910	0.979

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, ridge, lasso, elastic net (EN), bagging (BG), random forest (RF) and gradient boosting (GB). #var is the number of variables included in each dataset.

Table 9: Out-of-sample relative MSE for neural network.

Dataset	#var	#ensemble	NN_1	NN_2	NN_3	NN_4
$\mathcal{M}_1$	3	1	1.119	0.974	0.960	0.959
		5	1.030	0.965	0.955	0.956
		10	1.017	0.966	0.956	0.957
$\mathcal{M}_2$	5	1	0.913	0.900	0.907	0.905
		5	0.915	0.898	0.897	0.891
		10	0.914	0.898	0.897	0.892
$\mathcal{M}_3$	12	1	0.905	0.891	0.894	0.891
		5	0.891	0.882	0.885	0.885
		10	0.890	0.882	0.884	0.885

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for four neural networks (NN). #var is the number of variables included in each dataset. #ensemble is the number of combined neural network predictors.

20% to 25%:

Table 10: Out-of-sample relative MSE for linear and tree-based regression.

Dataset	#var	HAR-X	Ridge	Lasso	EN	BG	RF	GB
$\mathcal{M}_1$	3	1.000	1.001	1.007	1.002	1.168	1.020	1.066
$\mathcal{M}_2$	5	1.009	0.956	0.969	0.944	1.065	0.935	1.048
$\mathcal{M}_3$	12	0.993	0.919	0.944	0.912	1.004	0.903	1.016

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, ridge, lasso, elastic net (EN), bagging (BG), random forest (RF) and gradient boosting (GB). #var is the number of variables included in each dataset.

Table 11: Out-of-sample relative MSE for neural network.

Dataset	#var	#ensemble	NN_1	NN_2	NN_3	NN_4
$\mathcal{M}_1$	3	1	1.163	0.978	0.952	0.958
		5	1.013	0.969	0.953	0.958
		10	0.996	0.968	0.954	0.958
$\mathcal{M}_2$	5	1	0.921	0.900	0.901	0.895
		5	0.907	0.892	0.889	0.886
		10	0.908	0.891	0.888	0.885
$\mathcal{M}_3$	12	1	0.937	0.893	0.899	0.903
		5	0.898	0.876	0.883	0.884
		10	0.896	0.876	0.882	0.886

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for four neural networks (NN). #var is the number of variables included in each dataset. #ensemble is the number of combined neural network predictors.

A.2 Simulation of noise

We add pure noise to the information set in order to examine the robustness of the ML methods. In particular, we generate autoregressive noise with varying levels of persistence and scale it relative to the realized variance (before standardization) as follows:

$$u_t = \beta u_{t-1} + \omega \varepsilon_t, \tag{23}$$

where $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0,1)$ is white noise. We set $\omega^2 = \gamma^2 \sigma^2_{RV}(1-\beta^2)$ with

$$\sigma^2_{RV} = T^{-1} \sum_{t=1}^{T} \left(RV_t - \overline{RV} \right)^2, \tag{24}$$

and $\overline{RV} = T^{-1} \sum_{t=1}^{T} RV_t$.

The above configuration means that (in the stationary setting at least) $\text{var}(u_t) = \gamma^2 \sigma^2_{RV}$, so the noise variance is proportional to the volatility of the underlying stock and comparable across settings. We simulate from (23) where $\beta = [0.00, 0.50, 0.75, 0.90]$ with $\gamma = 0.25$ fixed and $\gamma = [0.25, 0.50, 1.00]$ with $\beta = 0.50$ fixed, yielding a total six different calibrations. It allows the noise to be both i.i.d. ($\beta = 0$) and dependent ($\beta \neq 0$), generating persistence in its time series.

We next simulate a set of noise variables that are conditionally heteroscedastic. In this setting, we construct covariates from the model:

$$u_t = \omega_t \varepsilon_t, \tag{25}$$

where $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0,1)$ and

$$\omega^2_t = c + \alpha u^2_{t-1} + \beta \omega^2_{t-1}. \tag{26}$$

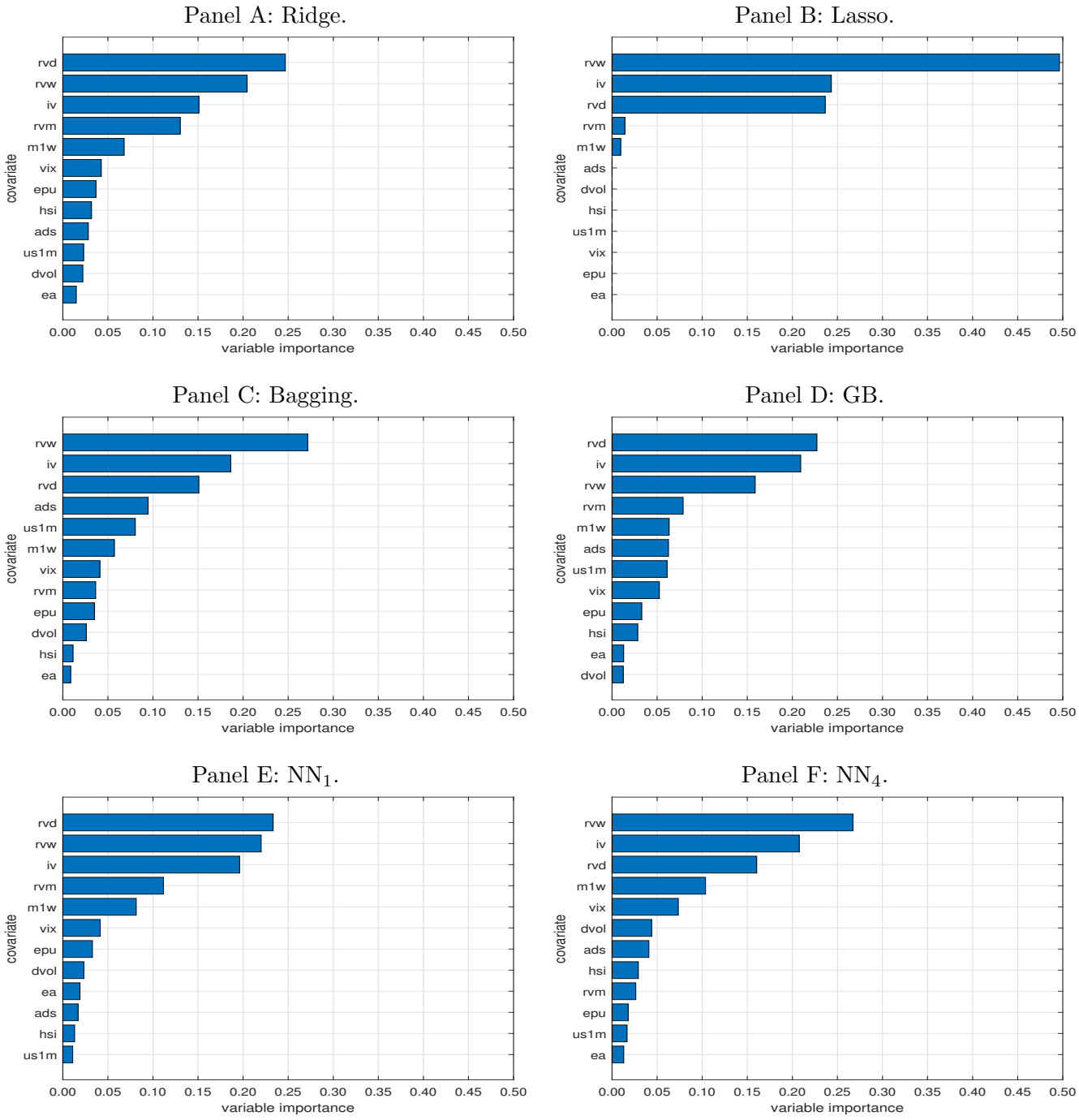
In line with empirical findings in the GARCH literature, we set $\alpha = 0.05$ and $\beta = 0.90$ to create strong volatility dependence. We select $c = \gamma^2 \sigma^2_{RV}(1-\alpha-\beta)$, so that the unconditional variance is $\text{var}(u_t) = \gamma^2 \sigma^2_{RV}$. As above, we take $\gamma = [0.25, 0.50, 1.00]$, yielding three combinations.

We include an equal number of noise terms from each model. This adds nine variables to $\mathcal{M}_4$ (one of each), 45 to $\mathcal{M}_5$ (five of each), and 90 to $\mathcal{M}_6$ (ten of each).

A.3 Additional VI measures for Apple

In the figure below, we report the VI measures for those forecasting models that were excluded from Figure 7 in Section 4.2 in the main text.

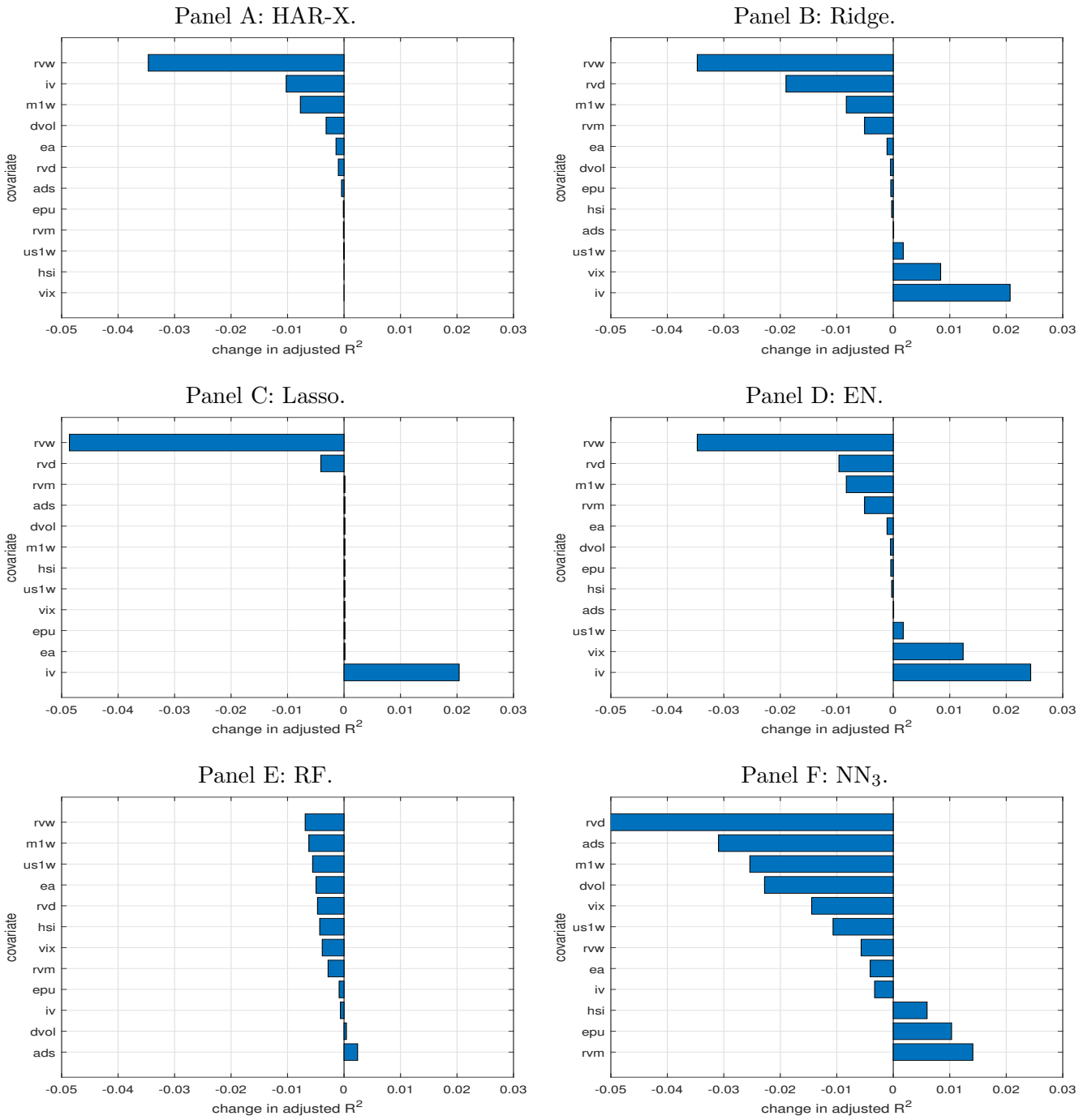
Figure 10: VI measure.



Note. We report the VI measure for each feature of ridge, lasso, bagging, NN_1, and NN_4 (NN_2 is omitted to conserve space), sorted according to the numeric value of VI. The figures are based on high-frequency data from Apple.

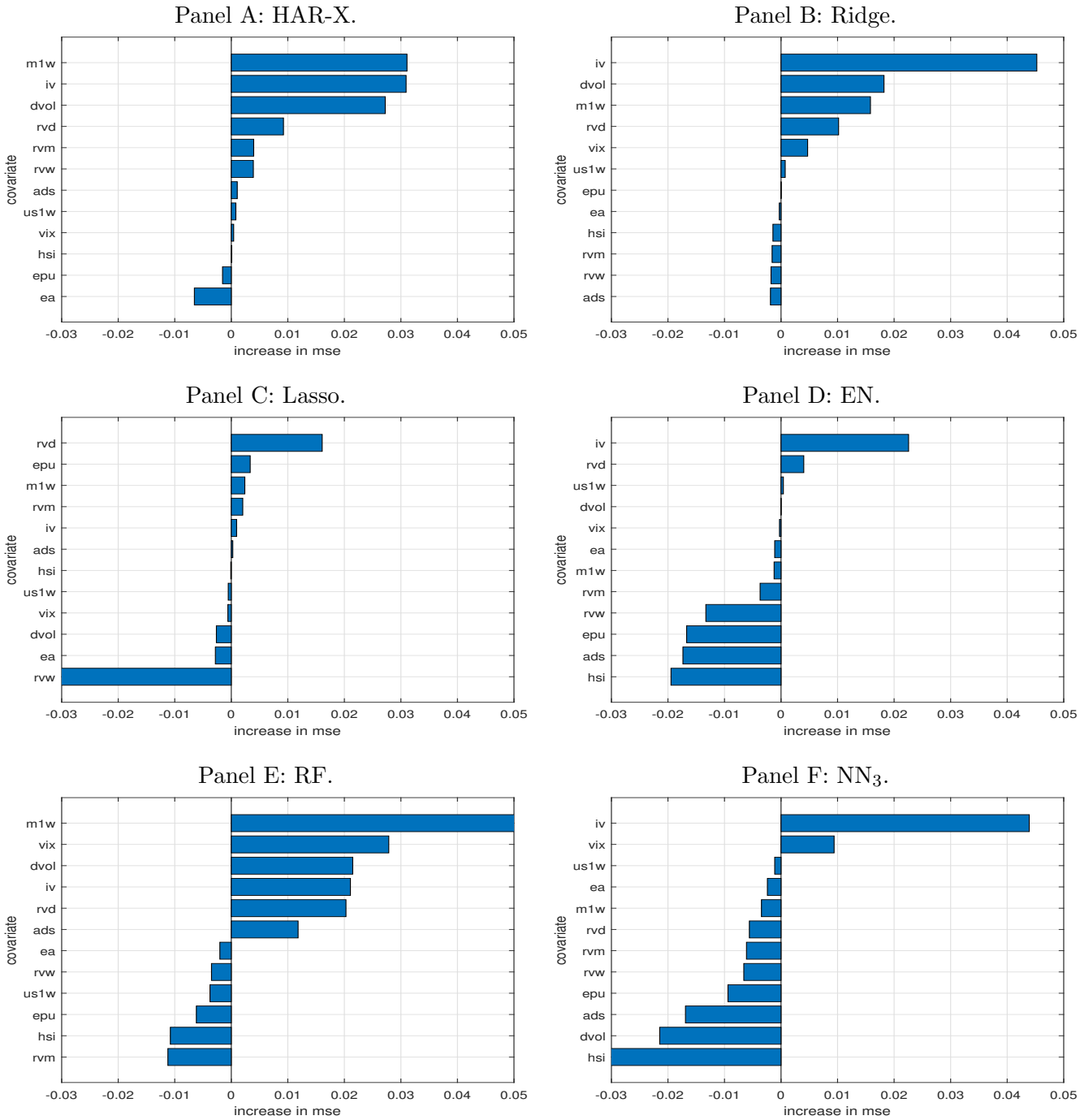
A.4 Alternative variable importance measures for Apple

Figure 11: In-sample change in adjusted R^2.



Note. We report the change in the in-sample adjusted R^2 associated with removing each feature one at a time in the HAR-X, ridge, lasso, elastic net, random forest and NN_3. The figures are based on high-frequency data from Apple.

Figure 12: Out-of-sample increase in MSE.



Note. We report the increase in out-of-sample MSE associated with removing each feature one at a time in the HAR-X, ridge, lasso, elastic net, random forest and NN_3. The figures are based on high-frequency data from Apple.

A.5 In-sample parameter estimates of the HAR and HAR-X model

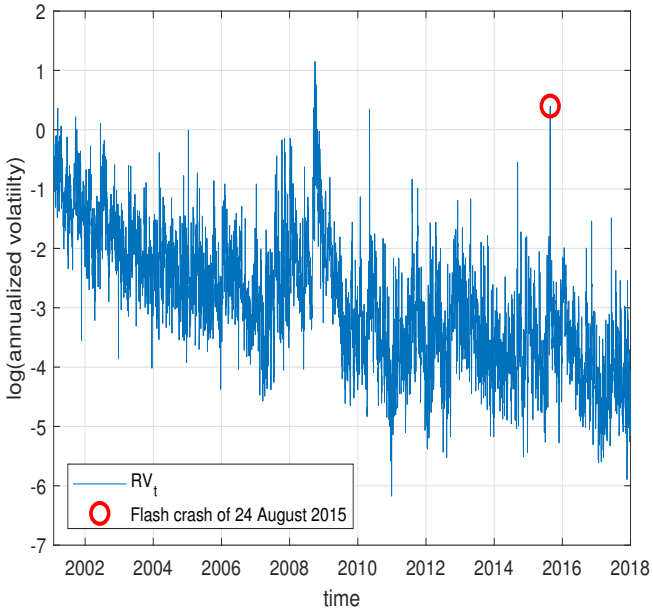
Table 12: In-sample parameter estimates of the HAR and HAR-X model.

	HAR	HAR-X	HAR-X
β_0	-0.039 (-1.212)	-0.009 (-0.283)	-0.011 (-0.340)
β_{RVD}	0.136 (1.545)	0.090 (1.098)	0.061 (0.685)
β_{RVW}	0.490 (3.520)	0.466 (4.174)	0.467 (3.244)
β_{RVM}	0.113 (1.565)		0.037 (0.524)
β_{IV}		0.168 (5.347)	0.161 (5.326)
β_{EA}		0.038 (4.921)	0.031 (3.916)
β_{EPU}			0.017 (0.686)
β_{VIX}			-0.020 (-0.491)
β_{US1M}			-0.014 (-0.927)
β_{HSI}			-0.005 (-0.148)
β_{M1W}		-0.082 (-3.423)	-0.083 (-3.681)
β_{DVOL}			0.034 (2.780)
β_{ADS}			-0.028 (-0.905)

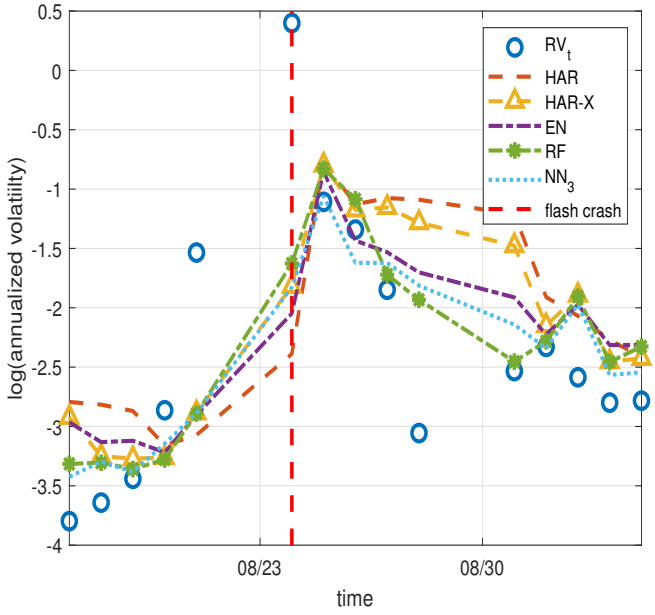
Note. The table reports in-sample parameter estimates of the HAR and HAR-X model for the one-day-ahead out-of-sample forecasting of Apple's return volatility. In parentheses, we report the t-statistic for testing statistical significance $H_0 : \beta = 0$. Standard errors are computed with a Newey-West correction. All the variables are standardized.

Figure 13: Flash crash of Apple stock price.

Panel A: Time series of RV_t.



Panel B: Prediction around the flash crash.



Note. The S&P 500 flash crash of 24 August 2015. Panel A displays the realized variance of Apple for the entire sample period. Panel B displays the actual and forecasted realized variance for all models around the flash crash of 24 August 2015.

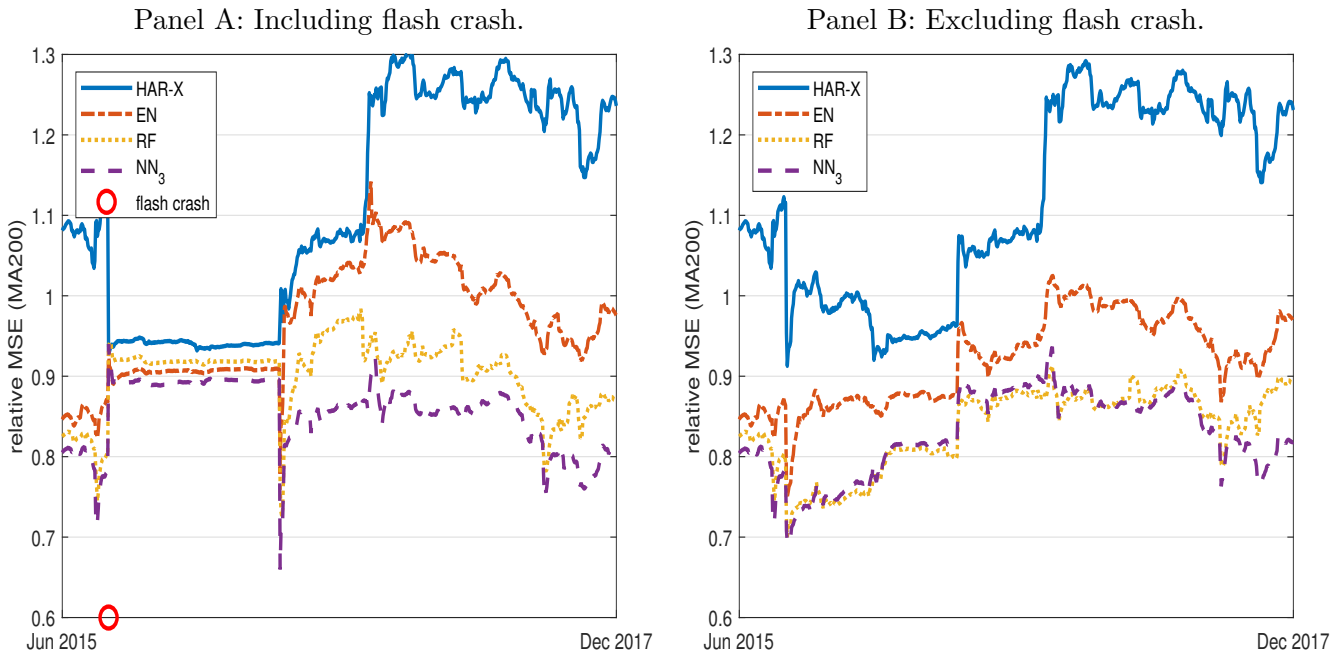
A.6 The flash crash of 24 August 2015

In this appendix, we investigate the behavior of the various models during an extreme market situation, namely the flash crash of 24 August 2015. In contrast to the financial crisis of 2008 and the flash crash of 6 May 2010, the former is part of the test set and, hence, our ML methods were not trained on this event.

The $\log(RV_t)$ time series of Apple is plotted in Panel A of Figure 13, where the flash crash is notably outstanding. As expected, neither of the models foresaw the event and end up predicting far too conservatively on the day, see Panel B in Figure 13. Our primary interest therefore concentrates on the aftermath of the flash crash. We observe that the HAR and HAR-X overestimate volatility in the subsequent days, where RV_t has largely reverted back to normal. In contrast, the ML methods appear to shrug the flash crash off as an outlier or at least attenuate its impact in their forecasts. In this aspect, the benefit of regularization is also evident, when we compare elastic net to the HAR-X model.

The flash crash has a large influence on the relative MSE loss, in part explaining why roughly equal performance is observed in the highest volatility decile in Figure 5 in the main text. This may raise concerns about the flash crash being the main culprit behind the superiority of the ML methods. However, we observe just the opposite in Panel A of Figure 14, where the relative MSE of the various models is calculated as a 200 day moving average. As evident, the flash crash (observation 55) dominates the relative MSE until it exits the rolling window, but it tends to align

Figure 14: Rolling out-of-sample relative MSE.



Note. The figure reports one-day-ahead out-of-sample forecast MSE relative to the HAR model calculated as a 200 day moving average. The out-of-sample window is rolled from June 2015 to December 2017. In Panel A the flash crash of 24 August 2015 is included, while it is excluded in Panel B. The dataset is $\mathcal{M}_3$.

the MSEs rather than disconnect them. In comparison, Panel B in the figure excludes the flash crash. As seen, the random forest and neural network continue to outperform the HAR even when the flash crash is discarded. Hence, the low average relative MSE of these models are due to consistent and sustained forecast accuracy, not luck on a single draw. This is further supported by Table 13 – 14, which reconstruct Table 2 – 3 in the main text but without data for 24 August 2015. In sum, without the flash crash we report an even further reduction of 6.2 and 4.5 percentage points in relative MSE for the random forest and NN_3 in the data-rich $\mathcal{M}_3$ environment.

Table 13: Out-of-sample relative MSE for linear and tree-based regression.

Dataset	#var	HAR-X	Ridge	Lasso	EN	BG	RF	GB
$\mathcal{M}_1$	3	1.000	1.010	1.015	1.010	1.204	1.068	1.077
$\mathcal{M}_2$	5	1.067	1.002	0.981	0.955	0.914	0.867	1.012
$\mathcal{M}_3$	12	1.053	0.960	0.943	0.902	0.894	0.844	0.970

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for HAR-X, ridge, lasso, elastic net (EN), bagging (BG), random forest (RF) and gradient boosting (GB). #var is the number of variables included in each dataset.

Table 14: Out-of-sample relative MSE for neural network.

Dataset	#var	#ensemble	NN_1	NN_2	NN_3	NN_4
$\mathcal{M}_1$	3	1	1.070	0.987	0.961	0.972
		5	0.997	0.976	0.963	0.965
		10	0.993	0.975	0.963	0.966
$\mathcal{M}_2$	5	1	0.879	0.866	0.848	0.843
		5	0.859	0.847	0.838	0.833
		10	0.855	0.844	0.837	0.834
$\mathcal{M}_3$	12	1	0.866	0.865	0.846	0.855
		5	0.836	0.832	0.837	0.840
		10	0.837	0.829	0.837	0.839

Note. The table reports one-day-ahead out-of-sample forecast MSE relative to the HAR model for four neural networks (NN). #var is the number of variables included in each dataset. #ensemble is the number of combined neural network predictors.

A.7 Hyperparameter tuning

The objective of the hyperparameters is to control the bias–variance tradeoff. The purpose of the majority of the hyperparameters is also to reduce the complexity of the model. The challenge is to do it in such a way that the model is capable of extracting relevant information from the observations and does not fit noise. There is a shortage of theoretical results in the literature on how to optimally select hyperparameters, so no superior strategy exists. In this brief appendix, we explain how we handle it.

We divide the dataset into a training, validation, and test set. In the training set, the model is estimated with various sets of hyperparameters. In the validation set, the estimated models are then compared based on their out-of-sample forecasts, and the optimal value of each hyperparameter is selected on this basis.[24] The test set serves as the true out-of-sample evaluation, where no estimation or tuning is conducted.

In the paper, we consider both a fixed and rolling estimation window. The less computational heavy is the fixed window. Here, the original data is split at the beginning, and the hyperparameters and weights are estimated once and employed throughout the test set. There is no re-estimating of the hyperparameters through time. The alternative is the rolling window, which allows for time-varying hyperparameters. Here, the hyperparameters are re-estimated each day with a fixed length of the training and validation. This approach entails inclusion of more recent observations during estimation, as past data are gradually excluded from the model.

Table 15 shows which hyperparameters are tuned (marked with an asterisk), and if so over which interval, and those that are set equal to the default value suggested by the original author. As apparent, we restrict tuning to a minimum.

[24]Standard k-fold cross-validation has not been conducted, as it violates the time-series structure in our data.

Table 15: Tuning of hyperparameters.

Regularization	BG	RF	GB	NN
$\lambda \in [10^{-4}, 10^{-1}]$*	min. node size = 5	min. node size = 5	depth $\in \{1, 2\}$*	learning rate = 0.001
$\alpha \in [0, 1]$*	trees = 500	trees = 500	trees $\in \{50, 100, \ldots, 500\}$*	epochs = 500
		feature split = $J/3$	learning rate $\in \{0.01, 0.1\}$*	patience = 100
				drop-out = 0.8
				initializer: Glorot normal
				Adam = default
				ensemble $\in \{1, 5, 10\}$

Note. The table presents the hyperparameters of the various ML algorithms. An asterisk indicates that the hyperparameter is tuned in the validation set. Bagging (GB) and random forest (RF): Breiman's RF with the default parameters based on the original Fortran code from Breiman and Cutler (2004). GB: The default parameters based on the implementation in Greenwell, Boehmke, Cunningham, and Developers (2019), who extended the AdaBoost algorithm of Freund and Shapire (1995). Neural network (NN): We employ the default parameters of the Adam optimizer suggested by Kingma and Ba (2014). The drop-out selection follows Goodfellow, Bengio, and Courville (2016).

A.8 Regularization of the neural network

Adaptive learning rate: To achieve fast convergence and locate a near-minimum function value, an adaptive learning rate is often imposed. The Adam optimizer, applied in this study, achieves that by shrinking the learning rate towards zero during training.

Drop-out: The idea behind drop-out is to temporarily remove neurons from an underlying layer, which in practice means that the output of that neuron is multiplied by zero with strictly positive probability (see, e.g., Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov, 2014). The drop-out rate is set to 0.8 following Goodfellow, Bengio, and Courville (2016). The drop-out is applied in the training of the network and all connections from the neurons are retained in the validation and test set.

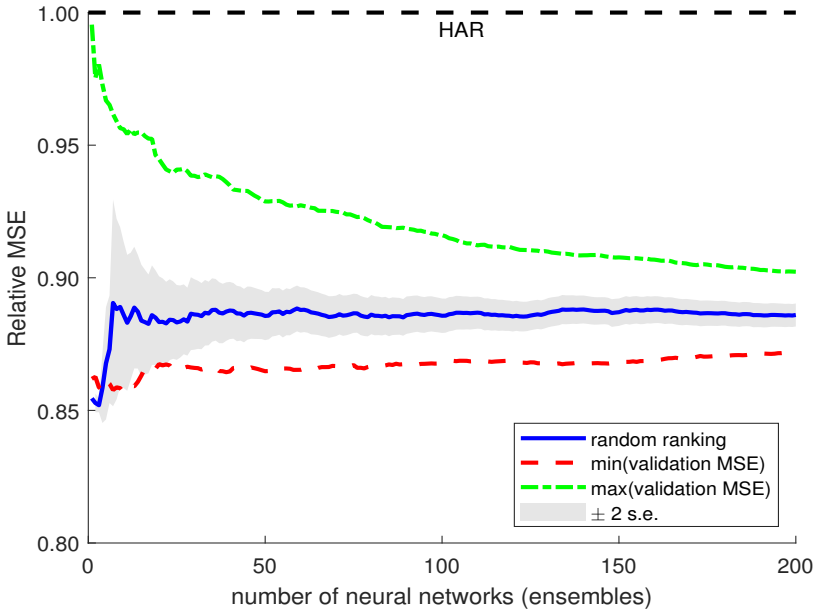
Early stopping: In early stopping, the goal is not to reach a local minimum of the training error, but to stop training the network when there is no improvement for some epochs in the validation error. To ensure sufficiently high patience, we set the patience to 100.

Ensemble: When conducting an ensemble, multiple neural networks are trained, and hereafter the prediction is constructed as an average across the ensemble. When initializing the weights, the Glorot (also called Xavier) normal initializer is chosen. The initialization draws samples from a truncated normal distribution centered on zero.

Initial seed: To ensure our results are not driven by randomness in the initialization, 400 independent neural networks with different seeds are trained and ranked by validation MSE. This allows to reduce the prediction variance induced by the stochastic features of the Adam optimizer (in the main text, we report the performance of the best network, plus an ensemble of the five and ten best networks found in the validation step).

As illustrated in Figure 15, a fast convergence to the actual performance is seen. Furthermore, changing the initial seed has a negligible impact. To further support this the figure also shows coverages, when ranking by highest and lowest MSE in the validation set. The relative MSE compared to the HAR model is below one for all initial values. The graph is based on high-frequency data from Apple, but the results are representative for other stocks in our sample.

Figure 15: Robustness check of initial seed for the neural network.



Note. On the y-axis, the graph shows the average MSE of NN_3 relative to the HAR model. The x-axis denotes the number of neural networks (ensembles) out of 400 that are averaged, when initial seeds are drawn at random (random ranking). The models are also sorted in order based on highest or lowest MSE in the validation set. The shaded area is ± two standard error bands of the random sort. The plot is based on high-frequency data for Apple, but the evolution is representative for other stocks in our sample.

Glossary

Activation function A function that takes the weighted sum of inputs (from the previous layer), adds a bias term, and then generates a nonlinear output, which serves as the input for the next layer. 7, 9

Adam Adam or "adaptive moments" is an adaptive learning rate optimization algorithm. It can be seen, as a combination of the RMSProp and momentum algorithm with a few distinctions. 8, 42, 43

Algorithm Is a finite sequence of well-defined functions or series of instructions used to solves a class of problems. 2, 5, 8, 10, 42

Boosting A machine learning technique that iteratively combines a set of weak learners to a strong learner. 1, 6

Epoch The number of times the algorithm is trained over the entire dataset. 42, 43

Feature Feature, explanatory variable, independent variable, and input variables are different terms for the same thing. 4, 10

Hidden layer A synthetic layer in a neural network. The layer is between the input layer (input features) and the output layer (the prediction). 7

Hyperparameters Hyperparameters are higher-level parameters of a model such as how fast it can learn (learning rate) or the complexity of a model. 4, 7, 10

Input layer The first layer in the neural network, which receives the input features. 7

Learning Rate The learning rate is the step size of each iteration. 6, 8, 42, 43

Momentum A gradient descent based algorithm which helps accelerate the convergence. 8

Nodes - Neural network A node or neuron in a neural network takes multiple input values and transforms them into an output value. The transformation is conducted through a nonlinear activation function. 7, 43

Nodes - Regression Tree *Root nodes*: Entry points to a collection of data. *Inner nodes*: A set of binary questions where each child node is available for every possible answer. *Leaf nodes*: Respond value if reached. 5

Output layer The final layer of a neural network. The layer containing the predictions. 7

Overfitting Overfitting occurs when your model starts learning the training data too well and begins to fit noise instead of information. 2, 3, 5, 8, 10

Regularization Regularization is a technique used for handling the overfitting problem. By adding a complexity term to the loss function, the sample variance of the model will decrease. 2, 3, 8, 12

RMSprop RMSProp (for Root Mean Square Propagation) is an adaptive learning rate algorithm. The learning rate is adapted for each of the parameters. RMSprop divides the learning rate by an exponentially decaying average of squared gradients. 8

Shrinkage Reducing the effect of sample variation by shrinking the coefficient towards zero. 4, 8, 12

Signal-to-noise ratio A measure of the amount of background noise with respect to the primary input. 1, 3, 12, 13

Sparsity Sparsity occurs when a large number of elements are set to zero in a vector or matrix. 4

Subset selection Selecting a subset of relevant variables. 4, 10, 12

Test Set A subset of data, at the end of the dataset, to assess the out-of-sample predictive power of the model. 9, 31, 41, 43

Training Set A subset of data used to fitting/training the parameters in the model. 9, 11, 41

Validation set A subset of data used to provide an unbiased evaluation of the current model specification. Often used for tuning the hyperparameters. 4, 6, 9, 41, 42

Weak learner A weak learner in machine learning, is a model that consistently beats a random guess. 6, 13