Journal of Empirical Finance 73 (2023) 251–271

Contents lists available at ScienceDirect

Journal of Empirical Finance

journal homepage: www.elsevier.com/locate/jempfin

Forecasting realized volatility with machine learning: Panel data perspective

Haibin Zhu [a], Lu Bai [b], Lidan He [c], Zhi Liu [b,d,*]

[a] Department of Statistics and Data Science, School of Economics, Jinan University, Guangzhou, China
[b] Department of Mathematics, University of Macau, Macao Special Administrative Region of China
[c] School of Mathematics and Statistics, Nanjing University of Information Science and Technology, Nanjing, China
[d] Zhuhai-UM Science and Technology Research Institute, Zhuhai, China

ARTICLE INFO

JEL classification:
C13
C14
G10
G12

Keywords:
Realized volatility
Panel data analysis
Machine learning
Forecasting

ABSTRACT

Machine learning approaches have become very popular in many fields in this big data age. This paper considers the problem of forecasting realized volatility with machine learning using high-frequency data. Instead of treating the realized volatility as a univariate time series studied by many existing works in literature, we employ panel data analysis to improve forecasting accuracy in the short term. We use six effective machine-learning methods for the realized volatility panel data. We compare our results with the traditional linear-type models under the same panel data framework and with the single time series forecasting via the same machine learning methods. The results show that the panel-data-based machine learning method (PDML) outperforms the other methods.

1. Introduction

Forecasting various quantities is an essential topic in finance. Among them, the quantity of the most interest is, of course, the asset price. Attempts have been made in this direction. See for instance Cochrane (2011), Welch and Goyal (2008), Campbell and Yogo (2006), Pettenuzzo et al. (2014) and references therein. However, many empirical studies (Chen et al., 2020; Gu et al., 2020) have shown that the consecutive asset returns are correlated very weakly and behave like the increments of a random walk, which usually yields a deficient predictive ability. Hence it is nearly impossible to predict short-term asset returns. The other quantity, the realized volatility, provides a possibility to explain the behavior of the markets. Brandt et al. (2010) found a strong negative correlation between idiosyncratic volatility and future asset returns. On the other hand, various stylized facts imply that the volatility is largely predictable.

Inspired by the seminal work of Engle (1982) and Bollerslev (1986), measuring and forecasting the volatility of assets has received growing attention. The consequent parametric models for the realized volatility have been considered by De Pooter et al. (2008), McAleer and Medeiros (2008), Clements and Krolzig (1998), Christiansen et al. (2012) and Guo et al. (2018), where they estimated asset volatility through a smooth transition or a threshold model. With the fast development of machine learning in recent years, the nonlinear relationship between variables can be successfully captured via variant neural networks. Kamijo and Tanigawa (1990), Donaldson and Kamstra (1996a,b) and Khan (2011) found that the artificial neural networks (ANNs) are helpful in forecasting volatile financial variables where the nonlinear dependence exhibits, for instance, the stock prices, exchange rate, etc. Moreover, combining neural networks with GARCH models provides further improvement to the forecasting of volatility, as

* Correspondence to: Faculty of Science and Technology, University of Macau, Avenida da Universidade, Room E11-3072, Taipa, Macao Special Administrative Region of China.
E-mail addresses: haibinzhu@jnu.edu.cn (H. Zhu), yb97466@umac.mo (L. Bai), hld@nuist.edu.cn (L. He), liuzhi@um.edu.mo (Z. Liu).

https://doi.org/10.1016/j.jempfin.2023.07.003
Received 2 April 2022; Received in revised form 12 February 2023; Accepted 26 July 2023
Available online 28 July 2023

studied by Hajizadeh et al. (2012), Donaldson and Kamstra (1997), Hu and Tsoukalas (1999) and Arneric et al. (2014), etc. Recent works by Rosa et al. (2014) and Miura et al. (2019) showed that the neural networks outperform the classical linear models in the out-of-sample forecast of realized volatility. We refer to Bucci (2018) for an overview of volatility forecasting.

In the classical linear models, only a small subset of the additional covariates are included in the forecasting model, partly because these methods rely on linear regression. Hence the methods break down when the model contains too many explanatory variables, which may be strongly correlated. A nonparametric neural network can overcome this approach, as studied by Bucci (2020), or conducting a variable shrinkage procedure to eliminate some less correlated variables first and then employ the neural networks to forecast, as investigated by He et al. (2021) and Christensen et al. (2021). Other machine learning techniques have also been employed in volatility prediction, namely, the Lasso, random forest, gradient boosting, etc. See Audrino and Knaus (2016), Audrino et al. (2020), Luong and Dokuchaev (2018) and Mittnik et al. (2015). Furthermore, machine learning techniques offer more flexibility in selecting the most efficient models.

The calculation of the realized volatility can be based on either low-frequency data (daily or lower) or high-frequency data (intraday, usually minutes level). The high-frequency data-based realized volatility, calculated as the summation of squared high-frequency returns, has been shown to be more efficient than that based on the low frequency returns. Lyócsa et al. (2021) showed that the high-frequency data-based realized volatility model outperforms the low-frequency data-based realized volatility model in the short-term forecast, while the difference in forecast accuracy is statistically indistinguishable in the long-term forecast (one month or longer). Moreover, it also separately defines the jump risk from the total risk. See Andersen et al. (2003) and Barndorff-Nielsen and Shephard (2004b).

Choosing the size of the training sample has been a widely discussed topic. See discussion in Rossi and Inoue (2012). Over a very long period, a time series with a suitable window size has to be used to train the model appropriately. However, it is well known that the model structure of financial markets rarely remains unchanged over a long time. For example, Inoue et al. (2017) developed a method for selecting the window size of forecasting with the structural changes in macroeconomic time series. Therefore, it would be desirable to have an extensive amount of data within a relatively short period. The cross-sectional time-series data meets the requirement. There are works making efforts in this direction. Han et al. (2018) and Wu et al. (2021) extended the machine learning methods for time-series data to the cross-sectional data. Han et al. (2018) implemented a combination of univariate forecasting by Lasso and elastic net regularization to predict cross-sectional stock returns. Wu et al. (2021) compared four workhorse approaches, namely, Lasso, gradient boosting, random forest, and deep neural network, in predicting hedge funds' performances.

Panel data have more variability and explore more issues than cross-sectional or time-series data alone. Moreover, the panel-data-based models, namely, a combination of time-series and cross-sectional models, accumulate more informative data over a short period, hence making the models more efficient. Baltagi (2008) provided a comprehensive overview of forecasting of the panel data. However, few works applied machine learning to the panel data framework. Recently, Chen (2021) used machine learning in the Boston housing panel dataset, Babii et al. (2022) introduced structured machine learning regressions for prediction and nowcasting with panel data. In this paper, we propose a new methodology to forecast the realized volatility, the panel-data-based machine learning approach (PDML), which uses various machine learning techniques to forecast the realized volatility panel where the cross-sectional realized volatilities are computed with high-frequency data. We aim to provide an efficient forecasting model for the realized volatilities with the dataset in a relatively short period.

The contributions of this paper are three-fold. First, we develop a panel-data-based machine learning method for forecasting daily realized volatility. Our models are built under a two-stage framework. In the first stage, we aim to select an optimal window size from an initial set of available choices based on the in-sample performance. In the second stage, we train the model associated with the optimal window size. Our results show that PDML outperforms classical linear models. Furthermore, our model framework can be embedded into any other machine learning model, making it very flexible for practical applications.

Second, time series models and panel models are often discussed separately. In the existing literature, there is a lack of comparison of their forecasting performance. For example, Fama and French (2020) compared cross-section and time-series factor models, considering size, value, profitability, investment, and momentum factors. They only investigated which model could provide better descriptions of average returns. Unlike their orientation, we want to compare their predictive ability. We design a comparison by letting the time-series model for each stock have the same training sample size as the panel-data-based model, in which window size is fixed. Our result indicates that panel-data-based and time-series models obtain comparable performance, but time-series models suffer from computational costs far exceeding that of the panel-data-based model. And more importantly, the two-stage PDML outperforms both in general.

Third, we are the first to organize a variety of high-frequency factors into panel models and to evaluate their usefulness. By shrinking the large-scale macroeconomic variables with machine learning methods, we detect some significant features in forecasting the future realized volatility. New high-frequency features, such as the realized semi-covariances, realized skewness and realized kurtosis, all exhibit significance in forecasting volatility. In practice, it will be very convenient to use these selected variables.

The rest of this paper is organized as follows. In Section 2, we introduce the basic notations for the panel data-based forecasting model. We review the machine learning methodologies in Section 3. In Section 4, we describe our datasets. Sections 5–7 contain the main results, including the feature importance and the evaluation of forecasting performance. We conclude the paper in Section 8.

2. Setting

We start with modeling the price process and introducing some basic statistics. Suppose our observation grid within a trading day is over the interval $[0, 1]$ with equally-spaced partition $\left\{t_j^n = \frac{j}{n} : j = 0, \ldots, n\right\}$. Denote $Y_T^{(i)}(t)_{\{t\in[0,1]\}}$ as the logarithmic asset price

for ith stock on day T defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$, following an Itô semi-martingale:

$$Y_T^{(i)}(t) = Y_T^{(i)}(0) + \int_0^t \mu_T^{(i)}(s)ds + \int_0^t \sigma_T^{(i)}(s)dW_T^{(i)}(s) + J_T^{(i)}(t),\ i = 1, \dots, N, \tag{1}$$

where $\mu_T^{(i)}(t)$ is a predictable and locally bounded drift process, $\sigma_T^{(i)}(t)$ is a càdlàg volatility term, $J_T^{(i)}(t)$ is a pure jump process, and $W_T^{(i)}(t)$ is standard Brownian motion. All processes mentioned above are adapted to the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$. The corresponding log-return $r_T^{(i)}(t_j^n)$ is calculated by $Y_T^{(i)}(t_j^n) - Y_T^{(i)}(t_{j-1}^n)$ for all T and any i.

We denote the ith stock's one-day realized volatility by $RV_T^{(i)}$ on day $T \geq 0$. Documented by Andersen et al. (2003) and further studied on its statistical inference by Barndorff-Nielsen and Shephard (2004b), the realized volatility is defined as

$$RV_T^{(i)} = \sum_{j=1}^{n} \left(r_T^{(i)}(t_j^n)\right)^2, \tag{2}$$

for $i = 1, \dots, N$. As the observation grid shrinks to zero, namely $n \to \infty$, realized volatility converges in probability to a quadratic variation process denoted by $\widetilde{\sigma}_T^{(i)}$:

$$RV_T^{(i)} \xrightarrow{p} \widetilde{\sigma}_T^{(i)} = \int_0^1 \left(\sigma_T^{(i)}(t)\right)^2 dt + \sum_{0\leq t\leq 1} \left(\Delta Y_T^{(i)}(t)\right)^2, \tag{3}$$

where $\Delta Y_T^{(i)}(t) = Y_T^{(i)}(t) - Y_T^{(i)}(t-)$.

In this paper, we are interested in forecasting $\widetilde{\sigma}_T^{(i)}$, which provides a prediction of stock price variation with two components: volatility risk and jump risk.

In the last decades, the most popular model applied to time-series forecasting for realized volatility is proposed by Corsi (2009), named the $HAR\text{-}RV$ model. They construct a time-series framework that contributes to a series of empirical applications in finance and economics literature when dealing with one-asset or index prediction. If we turn to a multi-assets forecasting problem, a panel model consisting of cross-sections is more favorable with its advantages. First, it provides a way to compare forecasting results among assets in the same cross-section. In volatility forecasting, this helps us select the model suitable for different volatility patterns under controlling other variables, i.e., different models for stocks with lower or higher volatility. Second, recent studies exhibit that loading on risk factors can be assumed constant during a short period but present a time-varying pattern over periods, see, Barroso et al. (2021) and Cheung (2023). Compared with time-series data, panel data leverage this advantage of requiring only a short period of historical data. Finally, while asymptotic theory for traditional methods requires a long period of historical data, the panel model considers only a sizeable cross-sectional dimension N with either a small or large time-series dimension. This superiority generally accommodates panel forecasting to various financial and economic problems, especially with poor data availability. In what follows, we will introduce our model settings of machine learning forecasting under a panel framework.

2.1. Panel-data-based model settings

Let $X_T^{(i)}$ be a p-dimensional vector of ith stock characteristics on day T for $i = 1, \dots, N$. A pooled model is used to illustrate the relation between future realized volatility and currently available characteristics:

$$RV_T^{(i)} = \alpha + X_{T-1}^{(i)}\beta + \epsilon_T^{(i)}, \tag{4}$$

where, α is a constant effect across all assets and cross-sections, β is a p-dimensional vector of factor loadings, and $\epsilon_T^{(i)}$ is an error term specific in a linear model. In particular, the constancy of α within one panel determines the "pooled" effect of our panel-data-based model.[1] While the linear relation is insufficient to describe the nonlinear relationship in the HAR-type model (Buccheri and Corsi, 2017), we apply some mainstream machine learning approaches to improve forecasting. The general expression, for asset i on cross-section T, follows the equation below:

$$RV_T^{(i)} = G(\alpha, \beta, X_{T-1}^{(i)}) + \epsilon_T^{(i)}, \tag{5}$$

where G depicts the true nonlinear relation between the response variable and features, $\epsilon_T^{(i)}$ is the error term. In following part of this article, on cross-section T, we denote RV_T as an N-dimensional vector collecting all target variables, X_T as a $N \times p$ matrix of input features with columns $X_T^{(1)}, \dots, X_T^{(N)}$. Now, our target is straightforward: estimate the function G before any cross-section T, so that the quadratic variation is predicted by $\mathbb{E}_T\left[RV_{T+1}^{(i)} \middle| X_T^{(i)}\right] = G(\alpha, \beta, X_T^{(i)})$. A pictorial illustration is shown in Fig. 1.

Suppose now we want to predict $RV_{T+1}^{(i)}$ by information at and before T. We compute the past daily realized volatility from $T - W_B + 1$ to T and all standardized input characteristics from $T - W_B$ to $T - 1$ within the same panel. Before model training, we need to select a window size W_B, which determines how many cross-sections up to T will be included in the training set. Model performance is usually sensitive to the choice of training window size, as pointed out by Rossi and Inoue (2012). To address this

[1] In fact, recent literature prefers to model the logarithmic realized volatility based on time-series studies (See Bucci (2020)). This is appealing for making data approximately Gaussian. However, we find that by cross-sectionally standardizing our econometric variables, our model virtually identical results. We refer authors to Section 4.5 in Christensen et al. (2021) for more illustrations.

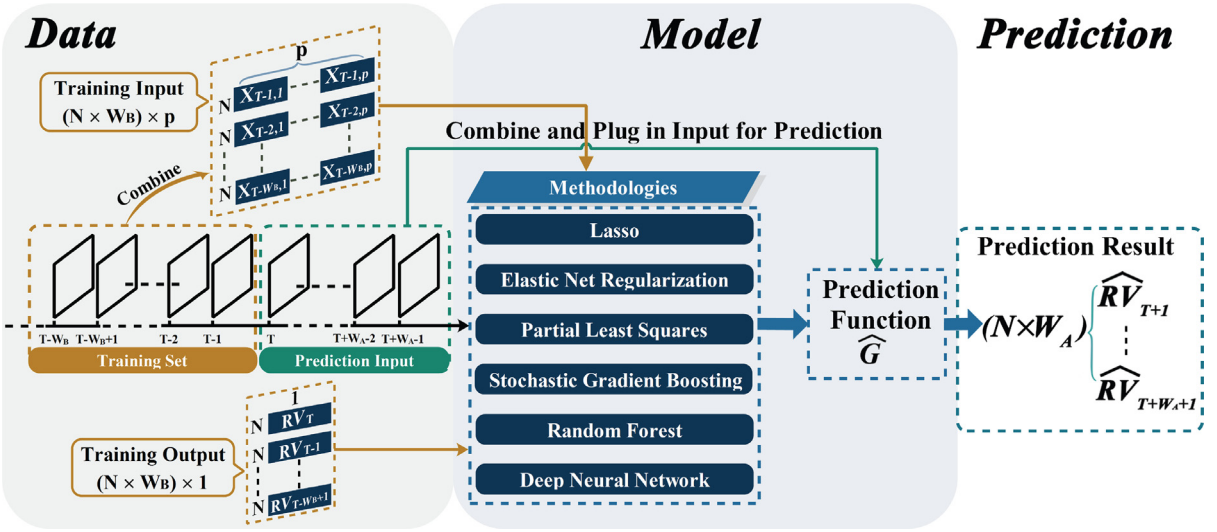

Fig. 1. Panel forecasting model structure. In the Data Section, we organize the training data into an $(N \times W_B) \times p$ matrix of explanatory input variables and an $(N \times W_B)$-dimensional vector of the target variable, which are then input into the Model Section in the middle. The Model Section is used for tuning hyperparameters and determining the final model, $\hat{G}$. Incorporated with the prediction input, an $(N \times W_A) \times p$ matrix of explanatory input variables, we finally output an $(N \times W_A)$-dimensional vector of prediction results for volatility.

issue, we design a two-stage framework for PDML. In the first stage, we select an optimal window size.[2] On which parameters we select, within each training set, a total of $N \times W_B$ training samples and p characteristics are used in fitting the function. That is, a panel of W_B cross-sections, each being represented by an $N \times p$ matrix, is stacked into a large $(N \times W_B) \times p$ matrix of explanatory input variables. We use the first $N \times (W_B - 1)$ samples as the first training and leave the last sample on $T-1$ to evaluate the model under different window sizes. Finally, the window with the smallest mean squared error is selected as the optimal choice, denoted by W_B^*.

In the second stage, we implement the training procedure by including available training samples dating back to $T - W_B^*$. We refer to each machine learning method and estimate G. We finally compute our forecasting $\widehat{RV}_{T+1}^{(i)} = \hat{G}(\alpha, \beta, X_T^{(i)})$ by a set of new input after cross-section T. The corresponding loss function is defined by

$$L(T, W_B, N) = \frac{1}{2} \sum_{j=1}^{W_B^*} \sum_{i=1}^{N} \left(RV_{T-j+1}^{(i)} - \hat{G}(\alpha, \beta, X_{T-j}^{(i)}) \right)^2 + \mathbf{P_T}, \tag{6}$$

where $L(T, W_B^*, N)$ is the loss including N assets from cross-section $T - W_B^*$ to $T-1$, and $\mathbf{P_T}$ is the penalty term at Tth cross-section. In particular, we define:

$$\mathbf{P_T} = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2, \tag{7}$$

where λ_1 and λ_2 are non-negative hyperparameters selected from cross-validation sets.[3] Notice that penalty exists only when either Lasso or ENR method is applied to training. Other approaches we will introduce in the next section set the penalty term to zero, i.e., $\lambda_1 = \lambda_2 = 0$.

The choice of hyperparameters and tuning parameters for each machine learning method is determined by designed validation. We divide all W_B cross-sections from the training panel into training subsets and validations under the framework of time-series blocked validation. Following previous studies, a validation set is used for tuning the hyperparameters conditional on which model is selected. We divide the training set into overlapped slices, each with a fixed-length training subset and validation set. For each fold at each iteration, we roll the slice forward by the length of the validation set, preventing testing on the same data.

Similar to Gu et al. (2020), we divide the entire dataset into a training set of the first 66 days and the remaining period for out-of-sample testing. We propose a rolling-forward forecasting framework in which our model is refitted on the last day of each month. On every refitted estimation, we use models to forecast daily volatility for the next month, where the number of trading days next month is denoted by W_A. That is, given models trained by data from $T - W_B^* + 1$ to T, we obtain realized volatility forecasts from $T+1$ to $T+W_A$. This, assuming models are stable over a short monthly period, allows for monthly forecasting. We recursively roll forward to the last day of next month and carry out our new forecasting of next-month daily realized volatility. Each time we refit, we maintain the training sample size and roll it forward to include the most recent month.

[2] Here, we choose the optimal W_B from either $5, 22, 44$, or 66, which corresponds to the trading period of one week, one month, two months, and one quarter, respectively.

[3] Usually, we set $\lambda_1 = \lambda\alpha$ and $\lambda_2 = \lambda(1-\alpha)$, where λ is non-negative and α is a real number in $[0, 1]$.

3. Machine learning methodologies

We aim to compare panel forecasting performance by different machine learning methods, benchmarking against the performance of the traditional linear model, such as cross-sectional Ordinary Least Squares (OLS) and time-series HAR. The basic linear model is easy to implement and analyze. However, it is shown to be underperformed by many financial studies, as we mentioned in Section 1. Recently, widely-used machine learning methods have been summarized by Gu et al. (2020), Wu et al. (2021) and Bucci (2020) for predicting cross-sectional asset returns, cross-sectional fund performance, and time-series index volatility, respectively. We focus on those machine learning methods, which are applicable to a cross-sectional framework, and extend them to our panel data.

Machine learning methods in our panel forecasting model can be categorized into three types: Lasso, Elastic Net Regularization (ENR), and Partial Least Squares (PLS) as the first group of statistical learning methods, Random Forest (RF) and Stochastic Gradient Boosting (SGB) as ensemble learning methods, and finally, Deep Neural Network (DNN) with different layer structures as deep learning methods.[4]

We provide a brief explanation of those adopted machine learning methods. Statistical learning approaches aim to construct a significant subset of features by shrinking insignificant features to a small value (Lasso Tibshirani, 1996 and ENR Zou and Hastie, 2005) or implementing a dimension reduction (PLS Wold et al., 1984). The dynamic of forming a smaller subset of features reveals the feasibility and interpretability of feature selection. On the contrary, ensemble learning methods (RF Breiman, 2001 and SGB Friedman, 2002), instead of shrinking features, iteratively generate alternative models and combine multiple outputs of forecasting. The main advantage of ensemble learning is its stability, which is robust to outliers and rewards for outperformance in small sample problems. Neural Network is a term that refers to a variety of network-type statistical learning approaches that process input information by neurons associated with activation functions or other structural designs in hidden layers and produce outputs from the network. It has become popular among all study fields for its ability to learn the non-linearity and complex relationships. Moreover, Neural Network carries strong flexibility in that we could easily implement modification of network architecture, such as dropout layer and recurrent network. In this article, we use the traditional feed-forward network, which could already exhibit its superiority in forecasting.

We will introduce all machine learning methods in terms of their statistical models and how their tuning parameters are set. In what follows, we assume a set of standardized training data $(X_1, RV_2), (X_2, RV_3), \ldots, (X_{W_B}, RV_{W_B+1})$. The purpose of applying machine learning below is to find a model that can best describe relationships between target and input variables.

3.1. Lasso

Lasso, proposed by Tibshirani (1996), is a type of regression analysis method that performs both characteristics selection and regularization, which can quickly and effectively extract important characteristics by shrinking the coefficients of less important features toward zero. This shrinkage property is based on the nature of L_1-regularization. The Lasso problem can be explicitly written as:

$$\left(\hat{\alpha}, \hat{\beta}\right) = \underset{\alpha,\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{j=1}^{W_B} \sum_{i=1}^{N} \left(RV_{T-j+1}^{(i)} - \alpha - X_{T-j}^{(i)} \beta \right)^2 \right\}, \quad \text{subject to } \|\beta\|_1 \le C, \tag{8}$$

where $C \ge 0$ is a tuning parameter, controlling the intensity of shrinkage on factor loading β's. The Lasso problem can also be written in the equivalent Lagrangian form as

$$\left(\hat{\alpha}, \hat{\beta}\right) = \underset{\alpha,\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{j=1}^{W_B} \sum_{i=1}^{N} \left(RV_{T-j+1}^{(i)} - \alpha - X_{T-j}^{(i)} \beta \right)^2 + \lambda \|\beta\|_1 \right\}, \tag{9}$$

where the term associated with Lagrange multiplier λ is the regularization term. Learning from Eq. (9), it is obvious that the larger value of λ is selected, the stronger penalty the model suffers. Typically, we allow λ to vary from 0.01 to 1, and finally select from our validation subsets based on in-sample errors. To solve the Lasso problem, there are many methods, such as Grafting and Coordinate Descent, etc. In this paper, we apply the penalized maximum likelihood estimation. We refer readers to Section 18.4 in Hastie et al. (2009) for more details of the algorithm.

3.2. Elastic Net Regularization (ENR)

On the opposite, with Lasso eliminating features from the models, the linear model associated with ridge penalty averages high-correlated features and shrinks them to each other, which implies coefficients do not hit zero even with a large shrinkage effect. To incorporate both, Elastic Net Regularization (ENR), proposed by Zou and Hastie (2005), combines the L_1 and L_2 regularizations. It has a similar sparsity of representation with Lasso, and shrinkages the coefficients together as Ridge, indicating it could accomplish regularization and features selection at the same time. Furthermore, ENR encourages a group effect where strongly correlated

[4] We omit some methods here, such as Principal Component Regression (PCR) and Support Vector Machine (SVM), in that PCR has a similar performance as PLS in cross-sectional prediction (Gu et al., 2020) and SVM is shown to be equivalently effective as Lasso (Jaggi, 2013).

characteristics tend to be in or out of the model together. This characteristic is particularly useful when the number of characteristics is much greater than the number of observations (Zou and Hastie, 2005).

Based on Eq. (4), the ENR problem can be written in the following form:

$$\left(\hat{\alpha},\hat{\beta}\right)=\underset{\alpha,\beta}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_{j=1}^{W_B}\sum_{i=1}^{N}\left(RV_{T-j+1}^{(i)}-\alpha-X_{T-j}^{(i)}\beta\right)^2\right\},\quad \text{subject to } (1-\kappa)\|\beta\|_1+\kappa\|\beta\|_2^2\le C, \text{ for some positive C,} \tag{10}$$

where κ is a hyperparameter controlling the mixture degree between Lasso penalization and Ridge penalization. Typically, a larger κ brings out more characteristics of Lasso to the model and generates sparse factor loading. In general, the term $(1-\kappa)\|\beta\|_1+\kappa\|\beta\|_2^2$ is called the elastic net penalty, which is a convex combination of lasso penalty and ridge penalty. For any $\kappa\in(0,1)$, the model is a compromise between Lasso and Ridge. If $\kappa=1$, the Elastic Net penalty becomes a Ridge penalty, while $\kappa=0$ induces the Elastic Net penalty to become a Lasso penalty. In other words, the size of κ determines the property of the penalty term, which is critical in model fitting.

The equivalent Lagrangian form of the ENR problem follows:

$$\left(\hat{\alpha},\hat{\beta}\right)=\underset{\alpha,\beta}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_{j=1}^{W_B}\sum_{i=1}^{N}\left(RV_{T-j+1}^{(i)}-\alpha-X_{T-j}^{(i)}\beta\right)^2+\lambda\left[(1-\kappa)\|\beta\|_1+\kappa\|\beta\|_2^2\right]\right\}, \tag{11}$$

where λ is a tuning parameter that controls the overall degree of penalization. Identical to Lasso, the ENR problem is solved by maximizing the penalized likelihood functions. The tuning parameters κ and λ are determined by in-sample performance on validation subsets. In our ENR model training, we allow κ to be from 0.1 to 0.9 and λ to vary from 0.01 to 1.

3.3. Partial Least Squares (PLS)

Partial Least Squares (PLS) regression was proposed by Wold et al. (1984), which is regarded as a combination of principal component analysis, canonical correlation analysis, and multiple linear regression analysis (Geladi and Kowalski, 1986). It is effective when the number of features exceeds the available observations since the intention of PLS is to reduce the dimension of the characteristics in the regression problem. In addition, it implements dimension reduction by forming components that capture most of the information carried by independent variables. That is, constructing new independent explanatory variables can eliminate the influence of multi-collinearity on regression and facilitate the prediction of response variables.

For each predictor denoted by $X_{T,m}$, $m=1,\dots,p$, we first project the target variable RV_{T+1} on $X_{T,m}$ to compute the coefficient φ_m which represents how much importance does target load on mth feature. We repeat the calculation of φ_m through all p features. Wold's PLS algorithm refers to Wold et al. (1984), which is an equivalence to an optimization problem, where φ_m is obtained by solving

$$\max_{\varphi}\operatorname{Corr}^2(RV_{T+1},X_T\varphi)Var(X_T\varphi),\quad \text{subject to } \|\varphi\|=1,\ \varphi' X_T' X_T\varphi_j=0, \text{ for } j=1,\dots,m-1. \tag{12}$$

The predicted value of RV_{T+1} can be obtained by

$$\widehat{RV}_{T+1}=(X_T\Omega_{K,T})\hat{\beta}_K, \tag{13}$$

where $\Omega_{K,T}$ is a $p\times K$ matrix with columns $\varphi_1,\dots,\varphi_K$, $\hat{\beta}_K$ is a K-dimensional vector of PLS factor loading calculated by regressing of target RV_{T+1} on PLS response variable $X_T\Omega_{K,T}$.

In this paper, we set the number of PLS components, denoted by K_{pls}, from 1 to 10, which is a tuning parameter. Usually, $K_{pls}<p$ implies a dimension reduction effect, while $K_{pls}=p$ is equivalent to the Ordinary Least Squares.

3.4. Random Forest (RF)

Random Forest (RF) is an ensemble classification and regression technique proposed by Breiman (2001). This ensemble is composed of a certain number of trees, in which each tree is constructed by using an injection of randomness. Since RF can fit the complex relations among data, such as non-linearity, it captures the relationship between each feature (Segal, 2004). RF regression uses the resampling technique to extract multiple sample subsets from the original sample, and each subset is modeled by a tree. By averaging the prediction from all trees, the model returns the result of the prediction. The algorithm to implement random forest is stated below.

Step 1: Generate subsets for each tree by bootstrap sampling from the original training set.[5] In this step, the size of each subset is about 2/3 of the original training set. In our study, we include 500 trees for RF (i.e., $N_{tree}=500$), so 500 subsets were sampled in this step.

[5] The most used sampling method are: (a) sampling without replacement; (b) boosting based on replacement sampling; (c) bagging based on boosting (Breiman, 1996).

Step 2: Construct trees. In this step, the algorithm constructs N_{tree} trees to form a forest where each tree is allowed to grow without pruning. Specifically, this step has two important processes:

(a) Node splitting. This is the core process in this algorithm. Through node splitting, a complete tree can be formed. The generation of the branches of each tree is to select attributes according to a certain splitting rule. These rules include information divergence maximization, information divergence rate maximization, minimum Gini coefficient, etc. In our construction of RF, the tree-splitting criterion is choosing the input feature with the lowest mean squared error.

(b) Random input selection. Since only partial features are involved in calculating attribute indices by node splitting, features are randomly selected. This stage is the so-called random input selection, which could reduce the correlation between each tree and improve their individual prediction performance. Therefore, it further improves the performance of RF (Breiman, 2001). In this study, we used another approach to determine the number of features sampled for splitting at each node (m_{try}). We set the selection of this hyperparameter ranges from 1 to 10, where different m_{try} leads to different MSE. The final choices of m_{try} are selected and associated with the lowest MSE losses.

Step 3: Repeat step 2 until the number of trees has reached the threshold so that trees are sufficient to form a random forest. The final output of the algorithm is calculated by averaging the prediction results of all trees. For example, assume the output of kth tree on T is $\widehat{RV}^{k}_{T+1}$. Then the outcome of prediction $\widehat{RV}_{T+1}$ by RF is calculated as $\widehat{RV}_{T+1} = \frac{1}{N_{tree}} \sum_{k=1}^{N_{tree}} \widehat{RV}^{k}_{T+1}$.

3.5. Stochastic Gradient Boosting (SGB)

Stochastic Gradient Boosting (SGB), proposed by Friedman (2002), incorporates randomization into the gradient boosting procedure by randomly selecting a subsample used to fit the base learners from the entire training sample at each iteration. Under such a setting, the update of model approximation is also random in successive iterations, shown by Friedman (2002). Specifically, in each iteration of SGB, a tree is grown from a random subset that is selected from the original data by sampling without replacement. Trees are added until the total residual deviance based on the held data does not decrease and finally reaches the most suitable prediction model.

SGB concerns three important hyper-parameters: the learning rate that determines the contribution of each tree to the whole model, tree complexity that determines the sample number in each final node, and the number of trees. In this article, we set the learning rate to be either $\nu = 0.01, 0.05$ or 0.1, the maximum depth of each tree could vary from 1 to 10, and the number of trees K is set to be either 50, 100, or 150, which is also the number of iterations. All hyperparameters are chosen by random search on the validation sets. The algorithm is implemented by Gradient Boosting Machine proposed by Friedman (2001).[6] We briefly illustrate the SGB algorithm procedure.

Assume there are K weak learners in SGB, implying that the algorithm iterates K times, each containing a base learner denoted by $h(X_T; \gamma_k)$ with parameter γ_k. The result of weak learner $f(X_T)$ on panel T is calculated by:

$$f(X_T) = h(X_T; \gamma_0) + \sum_{k=1}^{K} \eta_k h(X_T; \gamma_k), \tag{14}$$

where η_k is an expansion coefficient, and $h(X_T; \gamma_0)$ is the initial learner. The kth learner was chosen at kth iteration with the first $k-1$ learners being fixed. The objective of this algorithm is to choose a prediction function by minimizing the loss criteria, denoted by ϕ, over the training dataset. In this article, we set ϕ as the root mean squared error function. Thus, the optimal prediction function $f(X_T)$ fitted on T is given by:

$$f(X_T) = \underset{f}{\operatorname{argmin}} \left\{ \sum_{j=1}^{W_B^*} \sum_{i=1}^{N} \phi \left(RV^{(i)}_{T-j+1}, f(X^{(i)}_{T-j}) \right) \right\}. \tag{15}$$

Then, consider $f(x)$ as a parameter and use gradient descent to get the solution. Let $f_{k-1}(X)$ represents the estimation of $f(X)$ at $(k-1)st$ iteration. Moving forward to the kth iteration, the algorithm randomly selects a subset from the original training dataset, in which samples being randomly selected are collected in set $\mathcal{T}_k$. For convenience, we define $\left(RV_{q,T+1}, X_{q,T}\right)$ as the qth subsample from the entire training sample, as an equivalence to $\left(RV^{(q-\tau N)}_{T-\tau+1}, X^{(q-\tau N)}_{T-\tau}\right)$ for $\tau = \lfloor \frac{q}{N} \rfloor$ and $q = 1, \ldots, W_B \times N$. The corresponding negative gradient $\widetilde{RV}_{j,T}$ is calculated by

$$\widetilde{RV}_{j,T} = -\frac{\partial \phi\Big(RV_{j,T}, f(X_{j,T})\Big)}{\partial f(X_{j,T})}, \quad j \in \mathcal{T}_k. \tag{16}$$

Estimating γ_k by minimizing the sum of squared error,

$$\gamma_k = \underset{\gamma}{\operatorname{argmin}} \sum_{j \in \mathcal{T}_k} \left(\widetilde{RV}_{j,T} - h(X_{j,T}; \gamma) \right)^2. \tag{17}$$

Then, η_k is obtained by minimizing the loss function,

$$\eta_k = \underset{\eta}{\operatorname{argmin}} \sum_{j \in \mathcal{T}_k} \phi\Big(RV_{j,T}, f_{k-1}(X_{j,T}) + \eta h(X_{j,T}; \gamma_k)\Big). \tag{18}$$

[6] Another choice of implementation is by XGBoost, which is faster but sacrifices part of accuracy in our example.

The updated estimate at kth iteration can be written as $f_k(X) = f_{k-1}(X) + \nu\eta_k h(X;\gamma_k)$, where ν is a shrinkage parameter. Finally, after K iterations, model could output the final result $f_K(X)$. If using the entire sample at each iteration, SGB is equivalent to GB.

3.6. Deep Neural Network (DNN)

Neural Network (NN) is the most famous machine learning method applied in financial studies. For example, Donaldson and Kamstra (1996a) applied the Artificial Neural Network to combine a set of time-series forecasting of market volatility. Zhou (2019) introduced a novel two-stage return prediction framework by combining a time-series LSTM with a cross-sectional Deep Neural Network, which outperforms the individual time-series LSTM model. More recently, Bucci (2020) reviewed some most commonly-used NN-type structures, such as FNN, ENN, and JNN, and compared their prediction performance on a time-series framework. The models we study in this article follow the simplest traditional feed-forward neural network with the learning technique of back-propagation. The more complicated structure could exhibit other useful properties in terms of different types of data.[7] Still, it may also easily overfit if the model complexity is overstated.

In this study, we consider Deep Neural Networks, which can include more than one hidden layer. Therefore, with p input units and N_h^l hidden units in each hidden layer, the input and output of each hidden layer for all cross-sections T can be expressed as

$$h_i^l(T) = \begin{cases} X_T w_{in}^l + \theta_1^l, & l = 1 \\ h_o^{l-1}(T) w_{in}^l + \theta_1^l, & l > 1, \end{cases} \tag{19}$$

$$h_o^l(T) = f(h_i^l(T) w_{out} + \theta_2^l), \tag{20}$$

where $l = 1, \ldots, L$ refer to the parameters at lth hidden layer, and w_{in}^l denotes the input weight matrix at lth hidden layer. Assume the number of hidden layer neurons N_h^l varies with the l, by (19), we have $w_{in}^1 \in \mathbb{R}^{p \times N_h^1}$ for the first hidden layer and $w_{in}^l \in \mathbb{R}^{N_h^l \times N_h^{l-1}}$ for successive layers. w_o^l is an N_h^l-dimensional vector of the output weights for all $l > 1$. θ_1^l and θ_2^l are the biases for lth hidden and output layers, respectively. f is the activation function; here, we adopt the logistic function. The output layer of this multi-hidden-layer neural network is fed by the output of the last hidden layer, and thus the final result is calculated by

$$\widehat{RV}_{T+1} = f(h_o^L(T)). \tag{21}$$

Recall our setting in Section 2, each cross-section contains N assets, i.e., $\widehat{RV}_{T+1} \in \mathbb{R}^{N \times 1}$, for any T. And the error function used to optimize the neural network in our study is the sum of squares. The DNN problem is solved by the resilient back-propagation with weight backtracking, where in each iteration, weight matrices are updated by the sign of partial derivatives of activation functions. Training will be terminated if either the gradient descent falls beneath the threshold or the training step reaches its maximum. In particular, we set the maximum step for the training to be 10^5 and the threshold as stopping criteria from being 0.01.

The multi-layer perceptron, compared with a single-layer case, evolves by adding more hidden layers, which is regarded as a better approach to fit a nonlinear model. In this article, we consider three constructions of multi-layer perceptron denoted by DNN1, DNN2, and DNN3, respectively, with one, two, and three hidden layers. Different from other literature, in which they fix the number of neurons for each layer, we allow the number of neurons at the first hidden layer to vary from 1 to 8, at the second hidden layer to vary from 1 to 4, and at the third hidden layer to vary from 1 or 2. Those hyperparameters are selected according to mean squared errors in validation subsets.[8]

4. Data and features description

For features constructed by intraday data, we use one-minute return data in the QuantQuote database from Jan 2, 1998, to Jun 30, 2020. Specifically, we treat the 1% trading days with the highest market volatility as outliers and remove them from our samples. Our dataset includes all stocks that have been listed in the S&P 500 index. In addition, we remove all stocks that have no trade on any day during this period. Finally, 392 stocks are used for analysis. Moreover, we remove all days around holidays to eliminate the so-called holiday effect. In Table 1, we provide a list of all tickers and corresponding average realized volatility in our sample.

To enlarge our sample size, as mentioned above, our window size can be up to 66 days, which allows us to consider the panel consisting of $66 \times 392 = 25\,872$ samples. As expected, the time-series dimension W_B is smaller than the cross-section dimension N. While a larger window size in the training data essentially improves the in-sample performance, a trade-off between the beta persistence property against the computational cost induces us to set a relatively small window size. In the finance literature, the usage of 5-day, 22-day, and 66-day periods are commonly adopted to include the past week's, month's, and quarter's information since each month contains 22 trading days. Here, our two-stage framework, considering W_B as a sequence of tuning parameters, provides this flexibility, which could automatically choose the optimal window size based on the in-sample performance. In addition, to avoid a lack of data, our rolling-fitted model started on Feb 28, 2001, and ended in the penultimate month over the data.

[7] For example, adding recurrence structure to model long-memory property by, i.e., FNN, ENN, and JNN.

[8] This flexible setting leads to a large improvement from setting a fixed number of neurons at each layer. We omit the empirical results of DNN with a fixed number of neurons here.

Table 1
List of tickers and average realized volatility of 392 stocks.

Ticker – Volatility (10^4)												
AAPL	6.12	CAH	2.76	EIX	4.84	IGT	5.86	MKC	2.18	PLD	4.15	TAP	2.74
ABC	3.28	CAR	22	EL	2.74	INCY	20.54	MLM	4.28	PNC	3.89	TEN	20.49
ABT	2.7	CAT	3.62	EMN	3.35	INTC	4.21	MMC	2.82	PNW	2.1	TER	8.77
ADBE	6.61	CB	3.92	EMR	2.95	INTU	7.1	MMM	2.07	PPG	2.65	TEX	8.07
ADI	5.91	CCK	11.4	EOG	5.67	IP	4.63	MO	3.04	PPL	2.8	TFX	3.14
ADM	6.34	CCL	4.89	EQR	3.3	IPG	5.72	MRK	2.45	PRGO	8.47	TGNA	5.64
ADP	2.54	CDNS	7.13	EQT	4.91	IRM	4.62	MRO	6.41	PSA	3.08	TGT	3.82
ADSK	7.14	CERN	6.62	ES	2.71	IT	6.71	MS	7.35	PTC	18.44	THC	12.97
AEE	2.11	CHD	2.42	ESS	3.27	ITT	3.18	MSFT	2.87	PVH	6.5	TIF	4.51
AEP	2.58	CHRW	4.66	ETFC	19.67	ITW	2.67	MSI	5.85	PXD	7.91	TJX	4.29
AES	9.91	CI	4.1	ETN	2.62	JBHT	6.92	MTB	3.3	QCOM	6.37	TKR	5.76
AFL	4.17	CIEN	16.73	ETR	2.47	JBL	7.69	MTD	4.05	R	4.75	TMO	4.35
AIG	12.11	CINF	4.5	EVRG	3.08	JCI	3.25	MTG	17.62	RCL	6.43	TROW	6.21
AIV	4.27	CL	1.96	EXPD	5.65	JEF	3.75	MTW	8.48	RE	3	TRV	3.25
AJG	2.59	CLF	12.25	FAST	5.42	JKHY	7.12	MU	11.15	REG	3.3	TSN	5.13
ALB	4.2	CLX	2.5	FCX	8.77	JNJ	1.59	MUR	4.88	RF	9.07	TT	3.82
ALK	6.67	CMA	4.67	FDX	3.22	JPM	4.77	MYL	5.5	RHI	4.21	TUP	6.68
ALL	3.86	CMI	4.7	FE	2.93	JWN	7.17	NBR	12.78	RIG	9.59	TXN	5.14
ALXN	14.19	CMS	4.07	FHN	7.15	K	2.42	NC	13.17	RJF	5.26	TXT	4.8
AMAT	6.65	CNP	6.74	FISV	5.03	KBH	8.84	NCR	4.72	RL	5.28	UDR	5.56
AMD	15.22	COF	6.17	FITB	8.31	KEY	7.67	NE	14.29	RMD	4.35	UHS	3.26
AME	2.71	COG	5.88	FL	9.05	KIM	4.78	NEE	2.16	ROK	4.19	UIS	20.72
AMG	4.81	COO	4.15	FLS	4.9	KLAC	8.02	NEM	6.15	ROL	3.83	UNH	3.39
AMGN	3.79	COP	3.23	FMC	3.6	KMB	2.01	NI	3.19	ROP	3.4	UNM	5.42
AMZN	9.44	COST	3.61	FOSL	11.08	KO	1.87	NKE	3.1	ROST	5.77	UNP	2.94
AN	8.5	CPB	2.74	FRT	3.56	KR	4.17	NOC	2.23	RRC	19.48	URI	8.91
ANF	8.79	CPRT	8.5	GD	2.42	KSS	4.69	NOV	7.85	RRD	8.15	USB	4.85
AON	2.98	CR	4.1	GE	3.69	KSU	5.95	NSC	5.56	SAN	6.28	VAR	3.12
AOS	3.67	CSCO	4.33	GHC	5.8	L	2.66	NTAP	11.88	SANM	29	VFC	3.26
APD	3.26	CSX	4.01	GIS	1.71	LB	6.4	NTRS	4.34	SBUX	5.56	VIAV	16.58
ASH	3.96	CTAS	5.61	GLW	8.36	LEG	4.21	NUE	4.57	SCHW	6.91	VLO	5.09
ATGE	5.81	CTB	8.71	GPC	3.1	LEN	8.07	NWL	4.36	SCI	25.26	VMC	4.17
ATI	9.54	CTXS	9.36	GPS	6.03	LH	14.59	NYT	6.04	SHW	4.13	VNO	3.45
ATVI	9.41	CVS	3.25	GT	8.25	LLY	2.74	O	3.77	SIVB	8.74	VTR	7.46
AVB	3.01	CVX	2.33	GWW	3.08	LM	4.98	ODP	19.31	SJM	2.12	VZ	2.61
AVY	2.87	D	2.12	HAL	6.96	LMT	3.33	OI	8.69	SLB	4.62	WAB	5.43
AXP	3.95	DD	3.97	HAS	4.9	LNC	6.51	OKE	3.84	SLG	5.25	WAT	3.8
AZO	3.68	DDS	10.57	HBAN	11.44	LNT	2.6	OMC	2.66	SLM	7.56	WBA	3.66
BA	3.76	DE	3.93	HD	3.46	LOW	3.89	ORCL	5.61	SNA	3.08	WDC	24.29
BAC	5.61	DGX	3.23	HES	4.68	LPX	11.54	ORLY	6.77	SNPS	5.54	WEC	2.55
BAX	2.27	DHI	8.24	HIG	6.88	LRCX	9.79	OXY	5.33	SNV	12.9	WELL	3.64
BBBY	7.59	DHR	2.21	HOG	4.89	LUV	6.53	PAYX	5.61	SO	2.64	WFC	4.63
BBY	5.61	DIS	3.83	HOLX	16.26	M	6.32	PBCT	4.71	SPG	4.24	WHR	4.36
BC	7.72	DISH	8.87	HON	3.54	MAA	3.48	PBI	4.97	SPGI	2.96	WM	3.9
BDX	2.86	DLTR	6.35	HP	6.81	MAC	5.94	PCAR	5.79	SPXC	5.06	WMB	10.25
BEN	4.07	DLX	5.3	HPQ	4.47	MAS	6.1	PCG	6.63	SPY	1.21	WMT	2.38
BGG	6.3	DOV	3.17	HRB	3.75	MAT	6.79	PCH	4.77	SRCL	7.57	WOR	9.16
BIG	7.23	DRE	5.53	HRL	2.62	MBI	12.47	PDCO	5.52	SSP	7.89	WY	3.76
BK	5.15	DRI	4.73	HSIC	5.87	MCD	2.64	PEG	2.6	STT	5.15	X	9.38
BLL	2.71	DTE	2.17	HST	7.3	MCHP	7.9	PEP	2.46	STZ	3.7	XEL	3.32
BMY	3.03	DUK	2.51	HSY	2.02	MCK	4.03	PFE	2.83	SWK	3.29	XLNX	7.86
BSX	6.29	DVA	7.67	HUM	8.65	MCO	3.99	PG	1.93	SWKS	15.56	XOM	2.31
BWA	3.81	EA	7.04	IBM	2.2	MDP	4.14	PGR	2.77	SWN	11.87	XRAY	4.14
BXP	3.19	ECL	2.53	IDXX	7.03	MDT	2.67	PH	3.48	SYK	2.48	XRX	8.85
C	7.48	ED	1.95	IEX	3.1	MGM	8.69	PHM	7.91	SYY	3.05	YUM	3.19
CAG	3.84	EFX	3.29	IFF	3.06	MHK	4.12	PKI	4.87	T	2.9	ZION	6.72

Now we turn to describe our choice of features. Many existing studies have already summarized a full set of characteristics. For example, Green et al. (2017) listed 94 characteristics predicting monthly stock return. Cederburg et al. (2020) selected portfolio by groups of features in Accruals, Intangibles, Investment, Momentum, Profitability, Trading, and Value. Regarding prediction realized volatility, Bucci (2020) included 11 monthly available features. However, when we turn to high-frequency forecasting, most features, especially some accruals and value features, are not updated on a daily basis. In light of our observation frequency, we review recent literature and summarize a list of available daily features as shown in Table 2, which is also accompanied by a Spearman correlation plot in Fig. 2. Most of them empirically proved their abilities to forecast realized volatility or other financial statistics using linear or nonlinear models. In each subsection below, we will explain why those groups of features are selected.

Table 2
List of input characteristics.

Feature category	Feature definition	Symbol
Variation and jump	Daily realized volatility	DRV
	5-Day realized volatility	WRV
	22-Day realized volatility	MRV
	Daily downside realized semivariance (Barndorff-Nielsen et al., 2008)	DNV
	5-Day downside realized semivariance (Barndorff-Nielsen et al., 2008)	WNV
	22-Day downside realized semivariance (Barndorff-Nielsen et al., 2008)	MNV
	Realized bipower downward variation (Barndorff-Nielsen et al., 2008)	NSJ
	Relative signed jump (Bollerslev et al., 2018)	RSJ
	Realized skewness	SK
	Realized kurtosis	KT
Performance and liquidity	22-Day Sharpe ratio	SR
	22-Day market beta	BETA
	Trading volume	V
	Two-day corrected bid–ask spread measure (Abdi and Ranaldo, 2017)	BAS
	Realized staleness (Kolokolov et al., 2020)	ST
Loadings	Loadings on VIX	LVIX
	Loadings on WTI crude oil return	LOIL
	Loadings on daily 3-month T-bill rate	TB

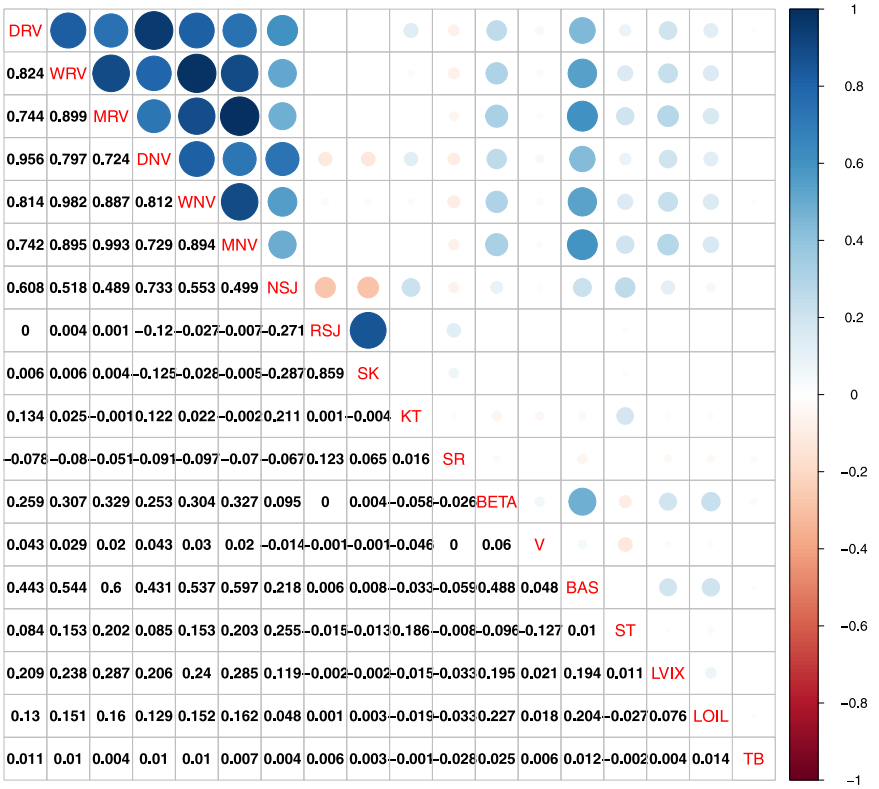

Fig. 2. Correlation matrix of features.

4.1. Variation and jump related features

The primary idea is to include statistics based on intraday returns. Our first choice of features comes from the most popular long-memory HAR model proposed by Corsi (2009), in which $RV^{(i)}_{T+1}$ is predicted by past daily realized volatility (with symbol **DRV**): $RV^{(i)}_{T}$, past-week average realized volatility (**WRV**): $\sum_{j=0}^{4} RV^{(i)}_{T-j}/5$ and past-month average realized volatility (**MRV**): $\sum_{j=0}^{21} RV^{(i)}_{T-j}/22$.

Despite a few limitations of HAR,[9] it is simple to be implemented and sufficient to depict the main variation in realized volatility processes. Barndorff-Nielsen et al. (2008) first derived statistical inference for realized semivariance and signed jump variation,

[9] For example, nonlinearity in features (See Buccheri and Corsi (2017)).

which is denoted by RS and SJ. Specifically, $RS_T^{(i)+}$ ($RS_T^{(i)-}$) denotes the positive (negative) realized semivariance for asset i on cross-section T:

$$RS_T^{(i)+} = \sum_{j=1}^{n} \left(r_T^{(i)}(t_j^n)\right)^2 \mathbf{1}_{\{r_T^{(i)}(t_j^n)\geq 0\}},$$

and

$$RS_T^{(i)-} = \sum_{j=1}^{n} \left(r_T^{(i)}(t_j^n)\right)^2 \mathbf{1}_{\{r_T^{(i)}(t_j^n)\leq 0\}}.$$

Based on the estimation of realized semivariance, signed jump variation for asset i on cross-section T, denoted by $SJ_T^{(i)}$, is defined by

$$SJ_T^{(i)} = RS_T^{(i)+} - RS_T^{(i)-}.$$

Patton and Sheppard (2015) found that both positive and negative realized semivariances improve the prediction of future realized volatility where the negative component plays a significant role. We symbolize the daily, weekly, and monthly average negative realized semivariance by **DNV**, **WNV** and **MNV** respectively, and are constructed similarly with DRV, WRV, and MRV described above.[10]

Since $RV_T^{(i)}$ and $RS_T^{(i)-}$ have already contained all information in $RS_T^{(i)+}$, we skip $RS_T^{(i)+}$, and modify $RS_T^{(i)-}$ to the Barndorff-Nielsen et al. (2008) realized bipower downward variation (denoted by **NSJ**) to capture the downside risk carried by negative jump variation. More empirical applications of jumps to volatility forecasting refer to Bu et al. (2023). Recall from Barndorff-Nielsen and Shephard (2004a), the realized bipower variation of asset i on cross-section T, capturing the continuous part of price variation, is defined as

$$BPV_T^{(i)} = \frac{\pi}{2} \sum_{j=2}^{n} \left|r_T^{(i)}(t_j^n)\right| \left|r_T^{(i)}(t_{j-1}^n)\right|.$$

The construction of NSJ is defined by:

$$NSJ_T^{(i)} = RS_T^{(i)-} - \frac{1}{2} BPV_T^{(i)}. \tag{22}$$

In addition, the information of upside risk is reflected in the relative signed jump (**RSJ**) proposed by Bollerslev et al. (2018), in which they found relative signed jump significantly predicts the cross-sectional asset returns. For asset i on cross-section T, denoted by $RSJ_T^{(i)}$, this estimator is constructed by:

$$RSJ_T^{(i)} = \frac{SJ_T^{(i)}}{RV_T^{(i)}}. \tag{23}$$

The last two return-based features are realized skewness (**SK**) and realized kurtosis (**KT**):

$$SK_T^{(i)} = \frac{n\sum_{j=1}^{n}\left(r_T^{(i)}(t_j^n)\right)^3}{\left(RV_T^{(i)}\right)^{\frac{3}{2}}} \quad and \quad KT_T^{(i)} = \frac{n\sum_{j=1}^{n}\left(r_T^{(i)}(t_j^n)\right)^4}{\left(RV_T^{(i)}\right)^2},$$

for asset i on cross-section T.

4.2. Asset performance and liquidity measure

Characteristics grouped in this subsection are related to assets' performance and their liquidity proxies. Most of them are commonly used in previous literature. For example, the 22-day Sharpe ratio (**SR**) and market beta (**BETA**) are two popular statistics to measure how the asset has performed in history. Besides, liquidity risk plays a critical role in idiosyncratic volatility modeling. An empirical illustration by Han and Lesmond (2011) points out that idiosyncratic volatility has minimal pricing ability by orthogonalizing some liquidity proxies. This implies that there potentially exist overlaps in volatility and liquidity, which may be modeled by virtue of machine learning. Here, we collect three representative liquidity-relevant characteristics: trading volume (**V**), Abdi's two-day corrected bid–ask spread measure (**BAS**)[11] and realized price staleness (**ST**).[12]

Trading volume and bid–ask spread are two well-known daily-available features depicting liquidity, while realized price staleness is a novel liquidity proxy. Lesmond et al. (1999) firstly proposed daily zero-return as a proxy for transaction cost. Upon high-frequency intraday data available, Bandi et al. (2017) and Bandi et al. (2020b) considered intraday realized price staleness as an evaluation of asset liquidity. Later, Buccheri et al. (2021) considered the average of idiosyncratic staleness across assets as a proxy for portfolio staleness. Realized staleness is connected to realized volatility prediction by enlarging the variation of realized volatility and by its positive correlation with liquidity.

[10] In detail, $DNV_T^{(i)} = RS_T^{(i)-}$, $WNV_T^{(i)} = \frac{1}{5}\sum_4^{j=0} RS_{T-j}^{(i)-}$ and $MNV_T^{(i)} = \frac{1}{22}\sum_{21}^{j=0} RS_{T-j}^{(i)-}$.

[11] Refers to Abdi and Ranaldo (2017).

[12] Construction of realized price staleness refers to Kolokolov et al. (2020).

4.3. Loading on common shocks

The last group of characteristics is influences carried out by some macroeconomic shocks. While all assets receive the same common information from the market, they react differently. Their sensitivity can be used as an idiosyncratic characteristic. See, for example, Fama and MacBeth (1973), Gu et al. (2020) and Wu et al. (2021). We select assets' realized volatility's loading on three representative macroeconomics shocks: VIX (**LVIX**),[13] WTI crude oil return (**LOIL**)[14] and U.S. daily 3-month T-bill rate (**TB**).[15] They are estimated from time-series least square regressions over the past month's daily data. We provide a short explanation of why they are serviceable in cross-sectional prediction.

The consideration of the Cboe Volatility Index (VIX) is straightforward. VIX, measured by options prices, represents the market's expectation of forward-looking market volatility. Wang et al. (2017) and Wang (2019) found that including the VIX component in the HAR-RV model can improve the prediction performance of realized volatility. The study of the relationship between crude oil price and stock price is a popular area of financial research. For example, Chang et al. (2013) applied various models to measure the conditional correlation between crude oil return and stock market index return. Wang et al. (2018) shows that the volatility of crude oil can predict the volatility of stocks, which encourages us to explore whether it could predict cross-sectional realized volatility. Finally, the 3-month Treasury bill (T-bill) rate is a type of short-term debt obligation reflecting many other macroeconomic shocks, such as investors' risk tolerance, inflation, and change in monetary policy. Hardouvelis (1987) treat the 3-month T-bill rate as one of fifteen representative macroeconomics variables, measuring the short-term interest rate negatively related to stock price's future response. The linear relation has been studied between the T-bill rate and stock return. See, for example, Addo and Sunzuoye (2013) find a long-term relationship between T-bill rate and stock returns. Nevertheless, the relation with realized volatility remains unclear. To fill the gaps, we explore its predictability through machine learning.

5. Out-of-sample forecasting performance

This section examines whether machine learning outperforms OLS and compares their performances. Two benchmark methods are considered. A prevalent linear model, Ordinary Least Squares (OLS), justifies the better performance of the machine learning methods. A panel-data-based HAR model (PHAR), proposed by Patton and Sheppard (2015), justifies the better performance of the panel-data-based model constructed in this paper.

For gauging forecasting performance, out-of-sample R^2 is an overarching priority. Campbell and Thompson (2008) proposed the out-of-sample R^2 benchmarking with historical mean, which is a widely-used criterion for measuring predictive accuracy, defined as

$$R^2_{mean} = 1 - \frac{\sum_{i=1}^{N}\sum_{\tau\in\Gamma}(RV_\tau^{(i)} - \widehat{RV}_\tau^{(i)})^2}{\sum_{i=1}^{N}\sum_{\tau\in\Gamma}(RV_\tau^{(i)} - \overline{RV}_\tau^{(i)})^2}, \tag{24}$$

where, Γ is the collection of entire cross-sections in test samples, $\overline{RV}_\tau^{(i)}$ is the historical average realized volatility for each asset i over each refitting training window corresponding to cross-section τ.

Another construction is suggested by Gu et al. (2020) in which the historical mean in the denominator is replaced by zero, named demeaning. This is a special case when predicting excess stock return since we assume stock return has zero means. This assumption supports evaluations by Eq. (24) because the historical mean of return is noisy and, thus, the over-evaluate performance of models. In contrast with a stock return, realized volatility, however, is positive over a period of the transaction. We need to benchmark forecasting results against a positive variable to assess predictive performance. Therefore, besides the choice of zero-benchmark out-of-sample R^2, we also propose an OLS-benchmark out-of-sample R^2 as

$$R^2_{OLS} = 1 - \frac{\sum_{i=1}^{N}\sum_{\tau\in\Gamma}(RV_\tau^{(i)} - \widehat{RV}_\tau^{(i)})^2}{\sum_{i=1}^{N}\sum_{\tau\in\Gamma}(RV_\tau^{(i)} - \widehat{RV}_{\tau,ols}^{(i)})^2}, \tag{25}$$

where $\widehat{RV}_{\tau,ols}^{(i)}$ is the ordinary least squares (OLS) forecasting with respect to the true $RV_\tau^{(i)}$ over the window after cross-section τ. R^2_{OLS} provides us an evaluation criterion for how much machine learning could over-perform or underperform the benchmark OLS. Better forecasting should carry out a lower mean squared out-of-sample error compared with OLS forecasting, which induces R^2_{OLS} getting closer to one. On the contrary, a negative R^2_{OLS} indicates that machine learning does not improve forecasting from a linear model. We report both R^2_{mean} and R^2_{OLS} trained by the entire sample in Table 3 respectively. Besides the forecasting performance of the whole period, we also report a grouped forecasting performance with respect to the volatility levels. We obtain the daily VIX data from YAHOO Finance and divide the whole period into four groups based on the 0.25, 0.5, and 0.75 quantiles of historical VIX levels. Then, we evaluate their performance within four subsamples.

Although out-of-sample R^2 provides an evaluation standard for comparison with benchmarks, there are some limitations. First, different from the prediction of other performance measures, such as asset return, predicting realized volatility is always involved by noisy proxy since the conditional variance of the asset is unobservable. For example, the term $\widetilde{\sigma}_T^{(i)}$ is a latent process, and using

[13] Data are available from the website of Yahoo Finance.

[14] Data are available from investing.com.

[15] Data are available from the website of FRED.

Table 3
Out-of-sample prediction R^2 with $W_B^* = 66$.

Criteria	Group	OLS	PHAR	PLS	Lasso	ENR	SGB	RF	DNN1	DNN2	DNN3
R^2_{mean}	All	0.069	0.214	0.151	0.185	0.216	0.217	0.153	**0.301**	0.262	0.126
	1 (Lowest)	0.288	0.300	0.338	0.314	0.355	**0.358**	0.173	0.236	0.141	0.068
	2	0.248	0.294	0.261	0.269	0.295	**0.418**	0.324	0.283	0.312	0.189
	3	−0.379	−0.301	−0.408	−0.368	−0.285	0.097	−0.020	**0.149**	0.096	0.034
	4 (Highest)	0.273	**0.481**	0.430	0.469	0.471	0.239	0.220	0.393	0.356	0.173
R^2_{OLS}	All	\	0.155	0.089	0.124	0.158	0.159	0.090	**0.248**	0.207	0.061
	1 (Lowest)	\	0.017	0.070	0.038	0.094	**0.099**	−0.161	−0.072	−0.206	−0.308
	2	\	0.060	0.016	0.026	0.061	**0.225**	0.101	0.045	0.084	−0.079
	3	\	0.056	−0.021	0.008	0.068	0.345	0.261	**0.383**	0.344	0.299
	4 (Highest)	\	**0.287**	0.216	0.270	0.273	−0.046	−0.072	0.165	0.115	−0.137

Note: Values in boldface represent the highest out-of-sample R^2 in corresponding groups. Symbol '\' indicates the result is undesired, in that prediction by OLS against itself is zero.

R^2 as a criterion suffers from the presence of noise. Second, out-of-sample R^2 does not directly select a subset of forecasting models that outperform others since several machine learning methods have been included.

To address the first issue, we introduce two loss functions used for constructing the test statistics. Shown by Patton (2011), MSE, and $QLIKE$ are two loss functions robust to a noisy volatility proxy,[16] in which they are estimated by

$$MSE = \frac{1}{N \times N_\Gamma} \sum_{i=1}^{N} \sum_{\tau \in \Gamma} (RV_\tau^{(i)} - \widehat{RV}_\tau^{(i)})^2, \tag{26}$$

$$QLIKE = \frac{1}{N \times N_\Gamma} \sum_{i=1}^{N} \sum_{\tau \in \Gamma} \left(\log(\widehat{RV}_\tau^{(i)}) + \frac{RV_\tau^{(i)}}{\widehat{RV}_\tau^{(i)}} \right), \tag{27}$$

where N_Γ denotes the number of element in Γ, the number of cross-section in test sample. The metric MSE depends on squared difference $(RV_\tau^{(i)} - \widehat{RV}_\tau^{(i)})^2$, and more sensitive to extreme observations in the sample. $QLIKE$, on the other hand, depending on standardized error $RV_\tau^{(i)}/\widehat{RV}_\tau^{(i)}$ tends to receive less effect from extreme value and penalizes more on under-prediction. This difference gives us a better understanding of why some methods endure relatively higher MSE but benefit from lower $QLIKE$ while others reverse.

In particular, when calculating MSE and $QLIKE$, we first annualize the predicted values and estimated realized volatility for the purpose of improving interpretation. Identical to the analysis of out-of-sample R-squared, we also divide training and test samples into subsamples by their historical realized volatility levels and carry out the results of MSE and $QLIKE$ from each group. The values of MSE and $QLIKE$ losses are reported in the first and third rows, respectively, in Table 4.

To tackle the second limitation, for a better comparison among methods, we also introduce a test named Model Confidence Set (MCS) by Hansen et al. (2011), which is improved from Superior Predictive Ability (SPA) by Hansen (2005). Compared with SPA, MCS has some attractive features. For example, MCS can identify informative data when uninformative data cannot distinguish models making MCS contain all available models. Another advantage of using MCS is that we do not need to specify a benchmark model so that more than one model could be the "best". Therefore, in this paper, we skip SPA and apply MCS to compare the forecasting performance of all methods.

Denote $L_{k,\tau}$ the loss function yielded by model k at cross-section τ (For example, we use MSE and $QLIKE$ here as loss functions), and M^0 the set containing a finite number of loss associated with models. The relative performance is defined by $d_{jk,\tau} = L_{j,\tau} - L_{k,\tau}$ and $\mu_{jk,\tau} = \mathbf{E}\left[d_{jk,\tau}\right]$ is finite for all $j,k \in M^0$ and for all $\tau \in \Gamma$. Using the same notation as in Hansen et al. (2011), the set of superior objects is defined by:

$$M^* := \{j \in M^0 : \mu_{jk,\tau} \leq 0\} \text{ for all } k \in M^0.$$

The null hypothesis is

$$H_{0,M} : \mu_{jk,\tau} = 0 \text{ vs. } H_{A,M} : \mu_{jk,\tau} \neq 0,$$

for all $j,k \in M \subset M^0$. Given a significant level α, we want to determine the set $\widehat{M^*_{1-\alpha}}$ containing all surviving objects where corresponding models have equivalent prediction performance and avoid being eliminated. The procedure of constructing $\widehat{M^*_{1-\alpha}}$ follows a standard MCS algorithm. Initially set $M = M^0$ containing all available objects. Then start the following two-step loop:

Step 1: Test the null $H_{0,M}$ associated with the current set M by an equivalence test δ_M.

Step 2: If $H_{0,M}$ is not rejected, set $\widehat{M^*_{1-\alpha}} = M$. Otherwise, eliminate an object from M by an elimination rule e_M, and repeat Step 1.

[16] Another detailed discussion on these two criteria refers to work by Bollerslev et al. (1994).

Table 4
MSE and $QLIKE$ loss functions with MCS test.

Criteria	Group	OLS	PHAR	PLS	Lasso	ENR	SGB	RF	DNN1	DNN2	DNN3
MSE Loss	All	0.071	0.061	0.064	0.061	0.059	0.049	0.049	**0.048**	0.059	0.086
	1 (Lowest)	0.013	0.013	0.012	0.012	0.012	**0.011**	0.012	0.014	0.017	0.031
	2	0.020	0.013	0.018	0.012	**0.012**	0.013	0.013	0.017	0.024	0.037
	3	0.142	0.128	0.142	0.139	0.126	**0.061**	0.063	0.064	0.074	0.085
	4 (Highest)	0.108	0.089	**0.083**	0.086	0.086	0.109	0.107	0.096	0.122	0.193
MSE P_MCS	All	0.062	0.947**	0.000	0.978**	0.982**	1.000**	1.000**	1.000**	0.000	0.000
	1 (Lowest)	0.129*	0.492**	0.676**	0.998**	0.351**	1.000**	0.616**	0.000	0.000	0.000
	2	0.000	0.380**	0.672**	1.000**	0.737**	0.609**	0.991**	0.000	0.000	0.000
	3	0.000	0.000	0.000	0.000	0.000	1.000**	0.932**	0.250**	0.002	0.003
	4 (Highest)	0.000	0.618**	1.000**	0.996**	0.999**	0.712**	0.946**	0.384**	0.000	0.000
QLIKE Loss	All	−1.655	−1.654	−1.660	−1.643	−1.651	**−1.697**	−1.663	−1.637	−0.260	−0.344
	1 (Lowest)	−1.948	−1.981	−1.887	−1.979	−1.977	−1.968	**−2.002**	−1.698	−1.000	−0.455
	2	−1.827	−1.909	−1.685	−1.907	−1.894	−1.902	**−1.923**	−1.569	−1.133	−0.208
	3	−1.615	−1.650	−1.626	−1.612	−1.619	**−1.707**	−1.695	−1.597	−0.602	−0.052
	4 (Highest)	−0.385	−1.064	−0.755	−1.040	−1.080	**−1.212**	−1.092	−0.592	2.267	−0.437
QLIKE P_MCS	All	0.000	0.000	0.000	0.000	0.000	1.000**	0.000	0.000	0.000	0.000
	1 (Lowest)	0.166*	0.000	0.173*	0.000	0.000	0.033	1.000**	0.000	0.000	0.000
	2	0.448**	0.013	0.421**	0.006	0.095	0.049	1.000**	0.000	0.000	0.000
	3	0.062	0.624**	0.240*	0.563**	0.660**	1.000**	0.996**	0.000	0.000	0.000
	4 (Highest)	0.000	0.019	0.000	0.000	0.000	1.000**	0.032	0.000	0.000	0.000

Note: Values in boldface represent the lowest losses in corresponding groups. The forecasts in $\widehat{M^*_{90\%}}$ and $\widehat{M^*_{75\%}}$ are identified by one and two asterisks respectively.

To calculate its p-value associated with object k in M, we adopt the test statistics $T_{R,M}$ from Hansen et al. (2011). In our empirical example, we highlight that $d^{(i)}_{jk,\tau}$ represents the relative performance associated with asset i, and calculate the aggregate relative performance by averaging across the cross-section τ. That is

$$T_{R,M} = \max_{j,k\in M} \frac{\frac{1}{n}\left|\sum_{\tau\in\Gamma} d_{jk,\tau}\right|}{\sqrt{\widehat{V_{jk,\tau}}}} = \max_{j,k\in M} \frac{\frac{1}{n\times N}\left|\sum_{\tau\in\Gamma}\sum_i d^{(i)}_{jk,\tau}\right|}{\sqrt{\widehat{V_{jk,\tau}}}},$$

where, $\widehat{V_{jk,\tau}}$ is an estimator of variance of $\frac{1}{n}\sum_{\tau\in\Gamma} d_{jk,\tau}$ which is estimated by a bootstrap of 5000 resamples (We refer to Bernardi and Catania (2018) about how to implement MCS algorithm).

5.1. Results

We start by showing the results of R^2. As how we define our sub-samples above, we list the results of R^2 with respect to four different quantiles of VIX levels in Table 3. Overall subgroups, DNN2 and DNN3, become unstable, which present to receive influence from extreme prediction errors easily. Compared with the DNN-type model, SGB and RF, resembling each other, perform well when volatility is relatively low. SGB, in particular, is more robust to low volatility scenarios when the market is sluggish, with R^2_{mean} remaining approximately 0.4 in both groups. Our second focus is ENR and Lasso, which both carry out the positive R^2_{OLS} across all subsample. This property is critical in risk management and empirical portfolio construction if volatility can be reliably predicted under any market conditions. Lasso, usually regarded as a similar model to ENR, does not present an equivalent performance as ENR. In addition, PHAR also outperforms OLS across all groups and surprisingly obtains the best performance when volatility is pushed up. This superior, however, sacrifices the forecasting accuracy when volatility is relatively low. The aggregate results of out-of-sample R^2 suggest DNN1 over the entire sample and ENR as the most stable approach.

Summarizing the above content, the following conclusions can be drawn. The first conclusion from the estimated out-of-sample R^2 by an entire sample is that all approaches, including both linear and non-linear models, have a better forecasting performance than the historical mean. Furthermore, all machine learning methods outperform the benchmark OLS model. Moreover, ENR, SGB, and DNN1 outperform the panel HAR model. In particular, DNN1 brings the best performance among all models by the entire sample, while SGB and ENR carry out the second and third-best performance associated with out-of-sample R^2. A longitudinal comparison of Table 3 reveals that ensemble methods have an outstanding performance during the period with low volatility but underperform others in subgroup 4. This result is related to the nature of ensemble methods to be robust to outliers. Another interesting finding is that the DNN-type model performs worst as the number of layers increases, i.e., DNN2 outperforms DNN3 but underperforms DNN1. A possible explanation may be a lack of features, leading to an overfitting issue with a more complex structure.

In conclusion, neural Network is famed for effectively processing predictions with an extremely large set of potential features. However, as data frequency increases, available features become fewer. For example, some accounting features are only reported monthly or even quarterly. This, to a certain extent, limits the performance of Multi-layer Neural Networks.

From Table 4, DNN1, ENR, and SGB, identical to the conclusion drawn from the analysis of R-squared, also exhibit outstanding prediction performance. For the entire sample, DNN1 outperforms other methodologies significantly with the lowest MSE loss, 0.048, and SGB receives the lowest $QLIKE$ loss, −1.697, respectively.

Table 5
Asset-level analysis of MSE and $QLIKE$ loss functions.

	Criteria	OLS	PHAR	PLS	Lasso	ENR	SGB	RF	DNN1	DNN2	DNN3
No. of assets ranked best	MSE	1	85	10	41	**141**	12	91	10	1	8
	QLIKE	41	9	71	1	10	20	**231**	9	0	0
No. of assets ranked top 3	MSE	9	295	57	170	**347**	56	201	35	5	1
	QLIKE	167	145	192	15	79	167	**374**	37	0	0
No. of assets ranked bottom 3	MSE	150	12	21	14	7	92	66	98	344	372*
	QLIKE	40	9	13	72	2	9	0	248	391	392*
No. of assets ranked worst	MSE	7	1	3	2	0	4	19	0	23	333*
	QLIKE	0	0	2	0	0	0	0	6	93	291*

Note: This table reports the number of times (over 392 assets) each method ranked best, top 3, bottom 3, and worst, respectively, in rows 1, 2, 3, and 4, associated with MSE and $QLIKE$ loss functions. Values in boldface imply the corresponding method has the best performance under certain criteria, while values assigned with one asterisk represent the corresponding worst performance. Losses are calculated for individual assets over the out-of-sample period.

Table 6
Different window sizes selected for each model and the average window sizes.

	Window size	OLS	PHAR	PLS	Lasso	ENR	SGB	RF	DNN1	DNN2	DNN3
No. of models	5	54	81	69	72	71	47	52	14	8	7
	22	59	49	32	23	37	44	51	59	36	41
	44	48	32	53	44	43	57	47	60	81	88
	66	70	69	77	92	80	83	87	98	106	95
Average window size		35.93	32.23	36.64	38.42	36.11	39.78	38.65	45.35	49.32	47.96

Note: This table reports the number of choices for each model for different window sizes and the average window sizes. Models are fitted monthly from Feb 28, 2001, to Feb 28, 2020. That is, a total of 231 months are included. The last row reports the mean value of the selected window size for each model.

Moreover, we can observe a significant tendency for MSE losses to become larger when the volatility levels increase. Over the entire sample, DNN1 outperforms other approaches and performs stable across all subgroups. SGB resembles RF but outperforms overall, carrying out the lowest MSE in groups one and three and the second-best performance over the entire sample. A notable phenomenon is that PHAR performs best in group 4, which coincides with the result in Table 3. For $QLIKE$ loss, SGB still leads the performance over the entire sample and becomes very stable.

Additionally, we exhibit p-values associated with losses MSE and $QLIKE$ in the second and fourth panels, where we choose the level of significance $\alpha = 0.10$. The interpretation of the result is simple and straightforward. A relatively high p-value, especially larger than α, indicates the corresponding object survives in $\widehat{M^*_{1-\alpha}}$. In our empirical investigation, we follow the setting of $\alpha = 10\%$ from Hansen et al. (2011). For the entire sample, the MCS based on MSE loss places PHAR, Lasso, ENR, SGB, and DNN1 in both $\widehat{M^*_{75\%}}$ and $\widehat{M^*_{90\%}}$, while MCS based on $QLIKE$ only place SGB. This result implies that SGB is the best predictive model robust to the noisy volatility proxy over our sample period and overall prediction error levels. When dividing assets into groups by historical level, SGB and RF are the only two methods placed in $\widehat{M^*_{75\%}}$ associated with MSE through all subgroups. On the contrary, considering QLIKE loss, DNN-type models fail to be comparable and even deteriorate with more hidden layers. The main reason is that DNN frequently under-predicts the true values.

Besides an aggregate result, we are interested in the prediction performance of each method at the individual asset level. We first calculate the forecasting MSE and $QLIKE$ average for each asset over the out-of-sample period. Then, among all methods, we output how many times each methodology ranked best, top 3, bottom 3, and worst out of 392 assets, as presented in Table 5. The asset-level analysis exhibits that ENR and SGB outperform others based on MSE and $QLIKE$ errors, respectively. According to $QLIKE$, DNN3 behaves worst because of the overfitting issue. For MSE loss, ENR exhibits superiority in individual asset forecasting, indicating its excellence in controlling errors induced by extreme observations. Surprisingly, the ensemble methods, SGB and RF, suffer abnormally higher mean squared errors at the asset-level analysis while slightly benefiting from evaluation by $QLIKE$. This anomaly is possibly owing to the inefficiency of ensemble methods to detect outliers, which could be shown by a relatively mild $QLIKE$. Recently, some comprehensive solutions, such as the isolation tree proposed by Cheng et al. (2019), aim to deal with outliers, providing a possible resolution to future investigation.

In Table 6, we show the results of optimal window sizes that each PDML model automatically selects. From left to right, we can see that the window size tends to become more prominent. This phenomenon reflects that the linear model prefers to use short-term historical data to ensure the constancy of factor loadings, while neural network models are eager for more training data to ensure training performance. In particular, the panel HAR model selects 87 cases with a window length of 5, which is not only the most frequent of all window size choices but also the highest among all methods. In sharp contrast to this is the neural network model, which does not choose a window size of 66 only in very few cases.

6. Panel versus time-series forecasting

The studies of prediction and forecasting in finance have been one of the most remarkable topics over the past decades. The innovation from time-series analysis to cross-sectional models started to receive attention mainly from Fama and MacBeth (1973),

Table 7
Comparison of time-series and panel data forecasting both with training size equal to 784 (time-series models are trained by 728-day data, and panel-data-based models are trained by 2-day with 392-stock data).

Model	Time-series models				Panel-data-based models			
	HAR_{ts}	ENR_{ts}	SGB_{ts}	DNN_{ts}	HAR_p	ENR_p	SGB_p	DNN_p
MSE loss	0.056	0.058	0.056	0.079	0.059	0.059	**0.055**	0.060
MSE p_{MCS}	1.000^{**}	1.000^{**}	1.000^{**}	0.000	0.990^{**}	0.990^{**}	1.000^{**}	0.830^{**}
QLIKE loss	**−1.659**	−1.656	1.624	−0.623	−1.508	−1.421	−1.618	−0.270
QLIKE p_{MCS}	0.000	0.000	0.000	0.000	0.000	0.000	1.000^{**}	0.000

Note: Values in boldface represent the lowest losses in corresponding groups. The forecasts in $\widehat{M^*_{90\%}}$ and $\widehat{M^*_{75\%}}$ are identified by one and two asterisks respectively.

and their relationship has been studied by, for example, Amihud (2002) and Fama and French (2020). While econometricians emphasize more on model explanation, little research makes an effort to compare their predictive abilities. Here, we make such a comparison.

In this experiment, our target is to compare the performance of time-series forecasts and panel-data-based forecasts under the same size as the training sample. Models used to train time-series data include HAR-RV and three commonly used models in panel forecasting, ENR, SGB, and DNN1. Comparison with other complicated time-series models, such as the nonlinear autoregressive exogenous model (NARX), long short-term memory (LSTM), and gated recurrent units (GRUs), is left for future research.

Distinct from the panel models, time-series models are fitted by one asset individually, but over a longer period. Recall in Sections 2 and 4, each panel-data-based model processes 66×392 samples with a blocked validation setting, whose training sample size is equivalent to fitting a one-asset time-series model by approximately 102-year historical data. This is infeasible in our model. To obtain an acceptable trade-off between data availability and prediction performance, we set two scenarios: time-series models fitted by 2×392 historical data and panel-data-based models with a fixed window size $W_B = 2$, both with a total of 784 training samples. To distinguish their notations, time series models are denoted with the 'ts' subscripts, and panel-data-based models are denoted with the 'p' subscript, respectively. For the time-series model, we treat the first 80% samples as training subsamples and the left 20% as validation samples. While for each panel-data-based model, we treat the first cross-section as a training subsample and the second cross-section as a validation sample. Identical to our previous setting, we refit models at the end of each month and make a W_A-day-ahead forecasting where W_A denotes the number of trading days in the next month. Once the forecasting results are obtained, we recursively forward to the last trading day of the next month.

Results are shown in Table 7. MSE and QLIKE losses imply that time-series models slightly outperform the panel-data-based models. However, when identifying the model confidence set that contains all models that perform equivalently, we find all panel forecasts stay in $\widehat{M^*_{90\%}}$ associated with MSE loss, significantly. The SGB_p even becomes the only model left in the set associated with QLIKE loss. This fact shows that panel-data-based models, though yield higher losses, still are comparable with time-series models by MCS test. However, the superiority in time-series models comes at cost of training time. Time-series forecasts are realized for each stock, while the panel forecast can synchronize all stock forecasts. Compared with the panel forecast, the time series forecast for all stocks will take N times more training time, where N is the number of stocks in the sample. In practical applications, this shortcoming involves high computational cost, impacting the timeliness of investment decisions if the number of assets considered explodes. Moreover, the scarcity of long-history high-frequency data makes this costly or even impossible for non-U.S. markets, such as most emerging markets, inducing that the panel-data-based model is appealing for empirical work. In other words, this demonstrates a clear advantage of panel prediction: We do not need a long-period training sample to get equivalent performance.

Another fact that highlights this advantage is the comparison of losses of panel-data-based models between Tables 4 and 7, which set $W_B = 66$ and $W_B = 2$, respectively. Intuitively, larger training samples tend to obtain better training results despite sacrificing computational efficiency. We find, surprisingly, reducing the time-series dimension of the panel does not worsen all model performances. In particular, ensemble methods benefit from the shortened time-series dimension, and DNN deteriorates significantly when the training window size is shortened. Statistical methods are stable under the change in window size. The main reason is due to a trade-off between information availability and factor loading stability. Coinciding with our discussion in Section 2, loading on risk factors should be considered time-varying over a long period, as shown empirically by Barroso et al. (2021). An over-extended training sample could interfere with the estimation of time-varying factor loadings, worsening the forecasting performance of panel methods. However, without a substantial amount of training data, machine learning, in particular, the DNN cannot handle massive parameters. Thus, the panel-data-based model should carefully choose the tuning parameters. In particular, our studies, comparing losses in Tables 4 and 7, imply that the PDML methods adaptive for different window sizes outperform time-series models.

7. Feature importance

Most features included in our models have been shown to be significant in predicting RV in literature. Efforts are devoted to the linear model. In this article, we innovatively include all daily updated features in the same model. Though high-frequency features are scarce, it gives insight into a question: Are they still significant in non-linear machine learning models? To address this issue, we discover their importance averaged all training samples among all methods. Note that we omit the feature importance analysis for HAR-type models because only three response variables are included.

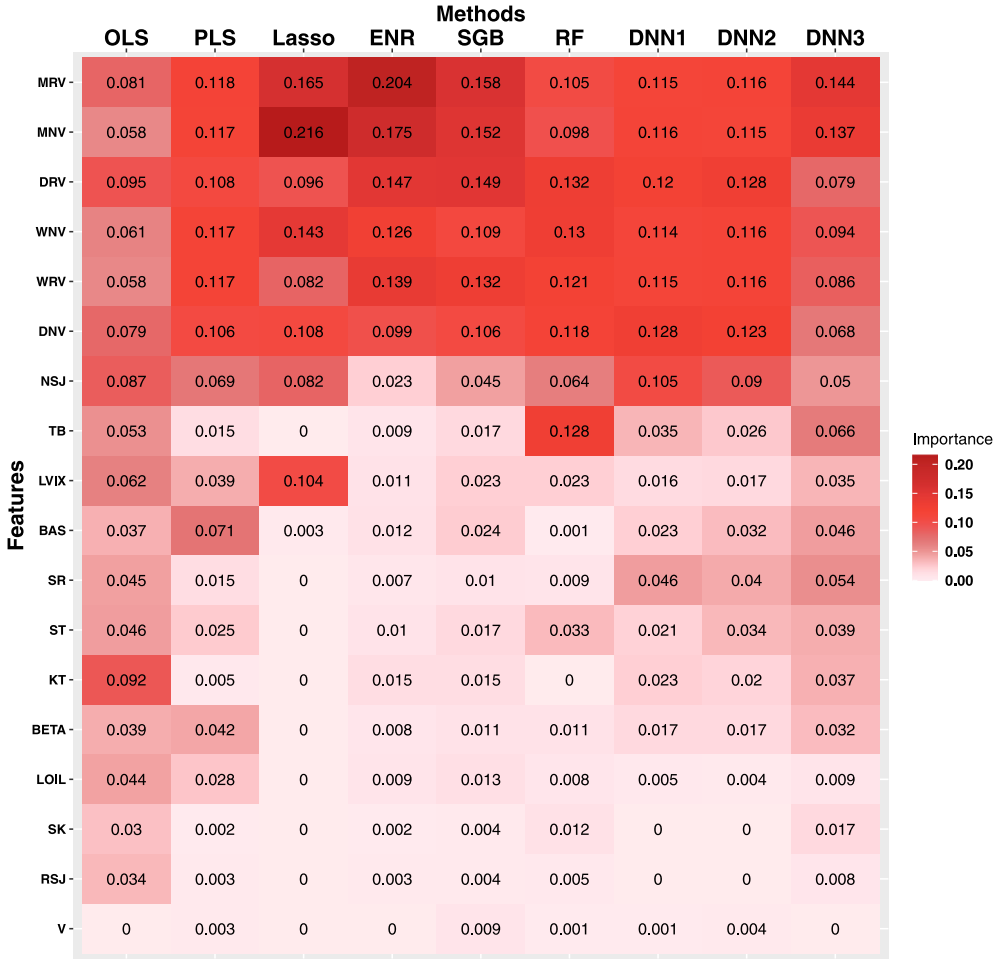

Fig. 3. Feature Importance corresponds to all methods, where the *y*-axis collects all features' names and the *x*-axis lists all methods. Their average importance sets the order of features among all methods, in which features on top carry out relatively higher importance. At the same time, those on the bottom have the least influence on models.

The ways to measure feature importance are various among all methods. The least complex model, OLS, measures feature importance by the absolute value of their individual t statistics. PLS, Lasso, and ENR, sometimes regarded as a group of similar models, measure feature importance based on the absolute value of coefficients. While Lasso and ENR use absolute values as measurement directly, PLS adopts the weighted sum of absolute coefficients in which weights are calculated by the amount of squared error across the entire training samples. For random forest regression (RF), we first calculate the forecasting error, which is MSE in our model, on each out-of-bag dataset. The error difference between the two instances is repeatedly calculated by all permutations and averaged over all trees. The feature importance is then measured by standardized average difference. Slightly different from RF, SGB measures importance based on forecasting error calculated using the entire training sample rather than the out-of-bag portion. Finally, for Neural Network-type models, feature importance is estimated by the MSE increment in the context of removing the feature from the model.

The result of feature importance is shown in Fig. 3 as an aggregate heat map and Fig. 4 as separated bar charts by methods. We first look at Fig. 3. The leftmost column lists the complete set of features in order of average importance in all methods, where features highlighted above have relatively higher importance. Our result demonstrates that the most highly important features are constructed based on intraday return processes related to recent volatility trends. Meanwhile, factor loadings on T-Bill rates and VIX also contribute to models comparatively. A surprising fact is that realized skewness and kurtosis are less significant than results from previous studies of the linear model. A possible explanation could be the non-linear depiction by other return-based features. Finally, volume is the most useless feature of all methods and is therefore placed at the bottom line.

In Fig. 4, we split features' ranking into sub-plots by each method to discover the features that models focus on. Generally, it is evident that Lasso performs shrinkage on features with fewer influences, generating sparse loading and inducing nine zero coefficients. On the contrary, ENR incorporating the property of Lasso with ridge penalties exhibits averaging effect and results in more non-zero features. Compared with Lasso and ENR, other methods have not presented a significant shrinkage effect in that their

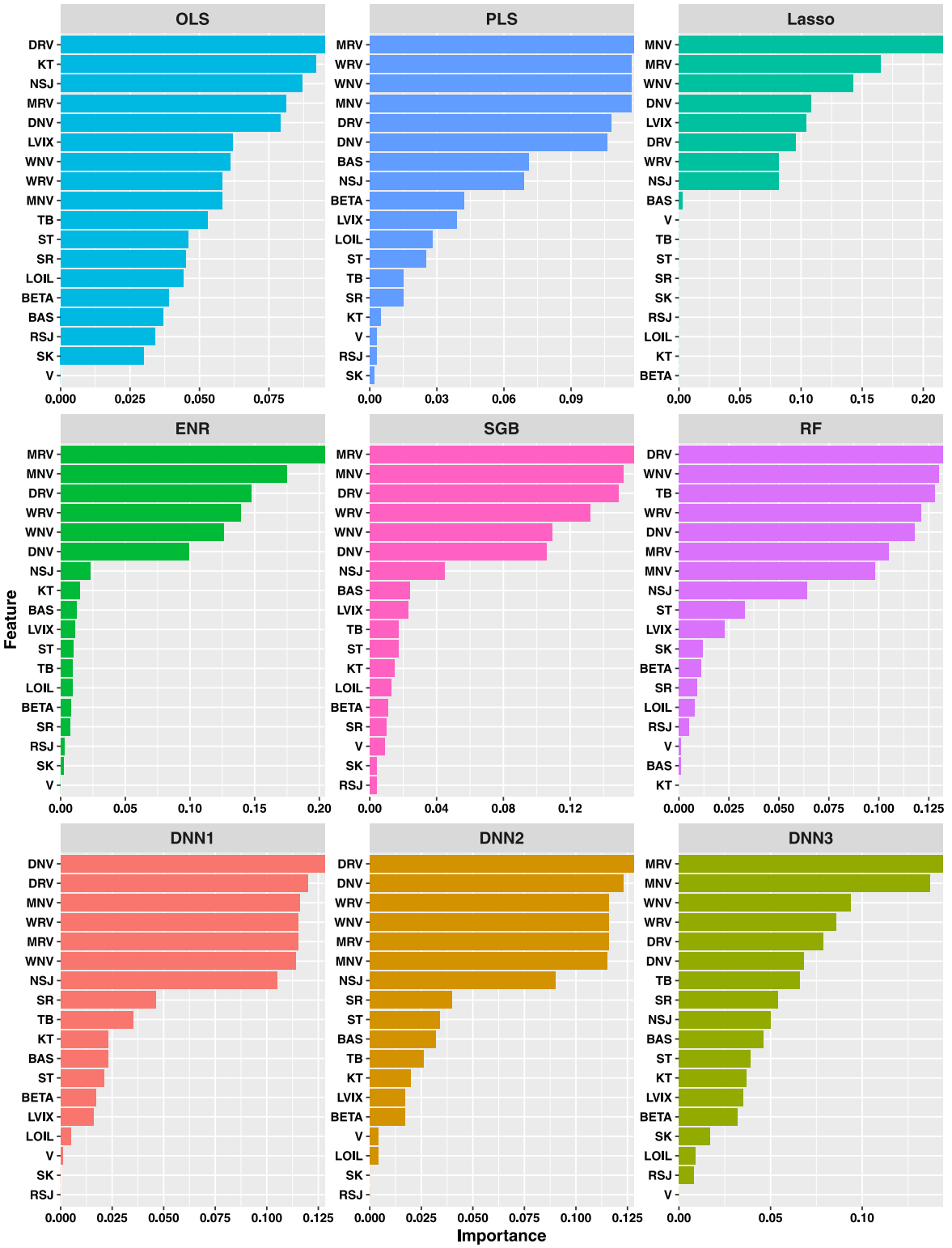

Fig. 4. Feature importance averaged over all training samples for all methods.

features at the half bottom have higher importance. In particular, OLS returns a model with almost equal feature importance and loads more on realized kurtosis, which is regarded as a negligible feature by other approaches.

8. Conclusion

In this paper, we propose a panel data forecasting framework for realized volatility by using high-frequency data. The multi-dimensional forecasting is conducted and rolled forward monthly, allowing for a constant factor loading over a relatively small time-series dimension, jointly shared by all assets. Our results are encouraging in that all machine learning techniques improve predictive power over classical linear models (e.g., ordinary least squares and heterogeneous autoregressive models) on a time-series basis or panel-data basis. Besides, panel data forecasting has presented its superiority in robustness to data availability. Elastic Net Regularization, Stochastic Gradient Boosting, and Neural Networks are three representative approaches that carry out an overall best forecasting performance.

We demonstrate that features with the highest importance are mainly constructed by high-frequency returns and strongly correlated with daily estimated realized volatility. Loadings on a daily 3-month T-Bill rate and loadings on VIX are the two most significant features constructed by not only return process, in that they are respectively regarded as two principal sources of market status: risk-free rate and market variation. Higher loading on significant common shocks usually implies a more vigorous intensity of assets' sensitivity to the market, leading to more frequent abnormal return variations when common shocks arrive. This evidence could contribute to future investigations of forecasting by high-frequency data.

CRediT authorship contribution statement

Haibin Zhu: Conceptualization, Methodology, Writing – original draft. **Lu Bai:** Visualization, Investigation. **Lidan He:** Software, Validation. **Zhi Liu:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Zhi Liu's research is supported by NSFC (No. 11971507), and The Science and Technology Development Fund (FDCT) of Macau (No. 0041/2021/ITP and 0079/2020/A2).

References

Abdi, F., Ranaldo, A., 2017. A simple estimation of bid-ask spreads from daily close, high, and low prices. Rev. Financ. Stud. 30 (12), 4437–4480.
Addo, A., Sunzuoye, F., 2013. The impact of treasury bill rate and interest rate on the stock market returns: Case of ghana stock exchange. Eur. J. Bus. Econ. 8 (2), 15–24.
Amihud, Y., 2002. Illiquidity and stock returns: cross-section and time-series effects. J. Financ. Mark. 5 (1), 31–56.
Andersen, T.G., Bollerslev, T., Diebold, F., Labys, P., 2003. Modeling and forecasting realized volatility. Econometrica 71 (3), 579–625.
Arneric, J., Poklepovic, T., Aljinovic, Z., 2014. GARCH based artificial neural networks in forecasting conditional variance of stock returns. Croat. Oper. Res. Rev. 5, 329–343.
Audrino, F., Knaus, S., 2016. Lossoing the HAR model: A model selction perspective on realized volatlity dynamics. Econom. Rev. 35 (8-10), 1485–1521.
Audrino, F., Sigrist, F., Ballinari, D., 2020. The impact of sentiment and attention measures on stock market volatility. Int. J. Forecast. 36 (2), 334–357.
Babii, A., Ball, R.T., Ghysels, E., Striaukas, J., 2022. Machine learning panel data regressions with heavy-tailed dependent data: Theory and application. J. Econometrics.
Baltagi, B.H., 2008. Forecasting with panel data. J. Forecast. 27 (2), 153–173.
Bandi, F.M., Kolokolov, A., Pirino, D., Reno, R., 2020b. Zeros. Manage. Sci. 66 (8), 3466–3479.
Bandi, F.M., Pirino, D., Reno, R., 2017. Excess idle time. Econometrica 85 (6), 1793–1846.
Barndorff-Nielsen, O.E., Kinnebrock, S., Shephard, N., 2008. Measuring Downside Risk-Realised Semivariance. CREATES Research Paper 2008-42.
Barndorff-Nielsen, O.E., Shephard, N., 2004a. Econometric analysis of realized covariation: high frequency based covariance, regression, and correlation in financial economics. Econometrica 72 (3), 885–925.
Barndorff-Nielsen, O.E., Shephard, N., 2004b. Power and bipower variation with stochastic volatility and jumps. J. Financ. Econom. 2 (1), 1–37.
Barroso, P., Boons, M., Karehnke, P., 2021. Time-varying state variable risk premia in the ICAPM. J. Financ. Econ. 139 (2), 428–451.
Bernardi, M., Catania, L., 2018. The model confidence set package for R. Int. J. Comput. Econ. Econom. 8 (2), 144–158.
Bollerslev, T., 1986. GeneGeneral autoregressive conditional heteroscedasticity. J. Econometrics 31, 307–327.
Bollerslev, T., Engle, R.F., Nelson, D.B., 1994. ARCH models. Handb. Econom. 4, 2959–3038.
Bollerslev, T., Li, S.Z., Zhao, B., 2018. Good volatility, bad volatility and the cross-section of stock returns. J. Financ. Quant. Anal. 55 (3), 751–781.
Brandt, M.W., Brav, A., Graham, J.R., Kumar, A., 2010. The idiosyncratic volatility puzzle: Time trend or speculative episodes? Rev. Financ. Stud. 23 (2), 863–899.
Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.
Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.
Bu, R., Hizmeri, R., Izzeldin, M., Murphy, A., Tsionas, M., 2023. The contribution of jump signs and activity to forecasting stock price volatility. J. Empir. Financ. 70, 144–164.
Buccheri, G., Corsi, F., 2017. HARK the SHARK: Realized volatility modeling with measurement errors and nonlinear dependencies. J. Financ. Econom. http://dx.doi.org/10.1093/jjfinec/nbz025.
Buccheri, G., Pirino, D., Trapin, L., 2021. Managing liquidity with portfolio staleness. Decis. Econ. Financ. 44 (1), 215–239.
Bucci, A., 2018. Forecasting realized volatility: A review. J. Adv. Stud. Finance 8 (2), 94–138.
Bucci, A., 2020. Realized volatility forecasting with neural networks. J. Financ. Econom. 18 (3), 502–531.

Campbell, J.Y., Thompson, S.B., 2008. Predicting excess stock returns out of sample: Can anything beat the historical average? Rev. Financ. Stud. 21 (4), 1509–1531.
Campbell, J., Yogo, M., 2006. Efficient tests of stock return predictability. J. Financ. Econ. 81, 27–60.
Cederburg, S., O'Doherty, M.S., Wang, F., Yan, X.S., 2020. On the performance of volatility-managed portfolios. J. Financ. Econ. 138 (1), 95–117.
Chang, C.L., McAleer, M., Tansuchat, R., 2013. Conditional correlations and volatility spillovers between crude oil and stock index returns. North Am. J. Econ. Finance 25, 116–138.
Chen, J., 2021. An introduction to machine learning for panel data. Int. Adv. Econ. Res. 27, 1–16.
Chen, L., Pelger, M., Zhu, J., 2020. Deep Learning in Asset Pricing. Working paper, https://arxiv.org/abs/1904.00745.
Cheng, Z., Zou, C., Dong, J., 2019. Outlier detection using isolation forest and local outlier factor. In: Proceedings of the Conference on Research in Adaptive and Convergent Systems. pp. 161–168.
Cheung, Y.L., 2023. Identification of time-varying factor models. J. Bus. Econom. Statist. 0 (0), 1–19.
Christensen, K., Siggaard, M., Veliyev, B., 2021. A Machine Learning Approach to Volatility Forecasting. CREATES Research Paper 2021-03.
Christiansen, C., Schmoling, M., Schrimpf, A., 2012. A comprehensive look at financial volatility prediction by economic variables. J. Appl. Econometrics 27, 956–977.
Clements, M., Krolzig, H.-M., 1998. A comparison of the forecast performance of Markov-switching and threshold autoregressive models of US GNP. Econom. J. 1, 47–75.
Cochrane, J., 2011. Presidential address: Discount rates. J. Finance 66, 1047–1108.
Corsi, F., 2009. A simple approximate long-memory model of realized volatility. J. Financ. Econom. 7 (2), 174–196.
De Pooter, M., Martens, M., Van Dijk, D., 2008. Predicting the daily covariance matrix for S&P 100 stocke using intraday data - but which frequency to use? Econom. Rev. 27, 199–229.
Donaldson, R.G., Kamstra, M., 1996a. Forecast combining with neural networks. J. Forecast. 15 (1), 49–61.
Donaldson, R.G., Kamstra, M., 1996b. A new dividend forecasting procedure that rejects bubbles in asset prices. Rev. Financ. Stud. 8, 333–383.
Donaldson, R.G., Kamstra, M., 1997. An artificial neural network-GARCH model for international stock return volatility. J. Empir. Financ. 4, 17–46.
Engle, R.F., 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica 50, 987–1007.
Fama, E.F., French, K.R., 2020. Comparing cross-section and time-series factor models. Rev. Financ. Stud. 33 (5), 1891–1926.
Fama, E.F., MacBeth, J., 1973. Risk, return, and equilibrium: Empirical tests. J. Polit. Econ. 81 (3), 607–636.
Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Statist. 1189–1232.
Friedman, J.H., 2002. Stochastic gradient boosting. Comput. Statist. Data Anal. 38 (4), 367–378.
Geladi, P., Kowalski, B.R., 1986. Partial least-squares regression: a tutorial. Anal. Chim. Acta 185, 1–17.
Green, J., Hand, J.R., Zhang, X.F., 2017. The characteristics that provide independent information about average us monthly stock returns. Rev. Financ. Stud. 30 (12), 4389–4436.
Gu, S., Kelly, B., Xiu, D., 2020. Empirical asset pricing via machine learning. Rev. Financ. Stud. 33 (5), 2223–2273.
Guo, T., Bifet, A., Antulov-Fantulin, N., 2018. Bitcoin volatility forecasting with a glimpse into buy and sell orders. In: 2018 IEEE International Conference on Data Mining (ICDM). IEEE, pp. 989–994.
Hajizadeh, E., Seifi, A., Fazel Zarandi, M., Turksen, I., 2012. A hybrid modeling approach for forecasting the volatility of S&P 500 index return. Expert Syst. Appl. 39, 431–436.
Han, Y., He, A., Rapach, D., Zhou, G., 2018. What firm characteristics drive us stock returns? Retrieved from: https://economics.indiana.edu/documents/workshop-papers/fall2018/what-firm-characteristics-drive-us-stock-returns.pdf.
Han, Y., Lesmond, D., 2011. Liquidity biases and the pricing of cross-sectional idiosyncratic volatility. Rev. Financ. Stud. 24 (5), 1590–1629.
Hansen, P.R., 2005. A test for superior predictive ability. J. Bus. Econom. Statist. 23 (4), 365–380.
Hansen, P.R., Lunde, A., Nason, J.M., 2011. The model confidence set. Econometrica 79 (2), 453–497.
Hardouvelis, G.A., 1987. Macroeconomic information and stock prices. J. Econ. Bus. 39 (2), 131–140.
Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.
He, L., Bucci, A., Liu, Z., 2021. Combining Dimensionality Reduction with Neural Networks for Realized Volatility Forecasting. Working paper, Available at SSRN 3824136.
Hu, Y., Tsoukalas, C., 1999. Combining conditional volatility forecasts using neural networks: An application to the EMS exchange rates. J. Int. Financ. Mark. Inst. Money 9, 407–422.
Inoue, A., Jin, L., Rossi, B., 2017. Rolling window selection for out-of-sample forecasting with time-varying parameters. J. Econometrics 196 (1), 55–67.
Jaggi, M., 2013. An equivalence between the lasso and support vector machines. In: Regularization, Optimization, Kernels, and Support Vector Machines. Available at arXiv: https://arxiv.org/abs/1303.1152.
Kamijo, K., Tanigawa, T., 1990. Stock price pattern recognition - a recurrent neural network approach. In: 1990 IJCNN Internation Joint Conference on Neural Networks. IEEE, San Diego, CA, USA, pp. 215–221.
Khan, A., 2011. Financial volatility forecasting by nonlinear support vector machine heterogeneous autoregressive model: Evidence from nikkei 225 stock index. Expert Syst. Appl. 3, 138–150.
Kolokolov, A., Livieri, G., Pirino, D., 2020. Statistical inference for price staleness. J. Econometrics 218 (1), 32–81.
Lesmond, D.A., Ogden, J.P., Trzcinka, C.A., 1999. A new estimate of transaction costs. Rev. Financ. Stud. 12 (5), 1113–1141.
Luong, C., Dokuchaev, N., 2018. Forecasting of realized volatility with the random forests algorithm. J. Risk Financ. Manag. 11 (4), 1–61.
Lyócsa, Š., Molnár, P., Vỳrost, T., 2021. Stock market volatility forecasting: Do we need high-frequency data? Int. J. Forecast. 37 (3), 1092–1110.
McAleer, M., Medeiros, M., 2008. A multiple regime smooth transition heterogeneous autoregressive model for long momory and asymmetries. J. Econometrics 147, 104–119.
Mittnik, S., Robinzonov, N., Spindler, M., 2015. Stock market volatility: Identifying major drivers and the nature of their impact. J. Bank. Financ. 58 (1), 1–14.
Miura, R., Pichl, L., Kaizoji, T., 2019. Artificial neural networks for realized volatlity prediction in cryptocurrency time series. In: Advances in Neural Networks. Springer International Publishing, Cham, pp. 165–172.
Patton, A.J., 2011. Volatility forecast comparison using imperfect volatility proxies. J. Econometrics 160 (1), 246–256.
Patton, A.J., Sheppard, K., 2015. Good volatility, bad volatility: Signed jumps and the persistence of volatility. Rev. Econ. Stat. 97 (3), 683–697.
Pettenuzzo, D., Timmermann, A., Valkanov, R., 2014. Forecasting stock returns under economic constraints. J. Financ. Econ. 114, 517–553.
Rosa, R., Maciel, L., Gomide, F., Ballini, R., 2014. Evolving hybrid neural fussy network for realized volatility forecasting with jumps. In: 2014 IEEE Conference on Computational Intelligence for Financial Engineering Economics. IEEE, London, pp. 481–488.
Rossi, B., Inoue, A., 2012. Out-of-sample forecast tests robust to the choice of window size. J. Bus. Econom. Statist. 30 (3), 432–453.
Segal, M.R., 2004. Machine learning benchmarks and random forest regression. In: UCSF: Center for Bioinformatics and Molecular Biostatistics. Retrieved from: https://escholarship.org/uc/item/35x3v9t4.
Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1), 267–288.
Wang, H., 2019. VIX and volatility forecasting: A new insight. Phys. A 533, 121951.
Wang, Y., Pan, Z., Wu, C., 2017. Time-varying parameter realized volatility models. J. Forecast. 36 (5), 566–580.
Wang, Y., Wei, Y., Wu, C., Yin, L., 2018. Oil and the short-term predictability of stock return volatility. J. Empir. Financ. 47, 90–104.

Welch, I., Goyal, A., 2008. A comprehensive look at the empirical performance of equity premium prediction. Rev. Financ. Stud. 21 (4), 1455–1508.
Wold, S., Ruhe, A., Wold, H., Dunn, W.J., 1984. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. SIAM J. Sci. Stat. Comput. 5 (3), 735–743.
Wu, W., Chen, J., Yang, Z., Tindall, M.L., 2021. A cross-sectional machine learning approach for hedge fund return prediction and selection. Manage. Sci. 67 (7), 3985–4642.
Zhou, B., 2019. Deep learning and the cross-section of stock returns: Neural networks combining price and fundamental information. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3179281.
Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B Stat. Methodol. 67 (2), 301–320.