CEWP 21-12

# Comparing Out-of-Sample Performance of Machine Learning Methods to Forecast U.S. GDP Growth

Ba Chu
Carleton University

Shafiullah Qureshi
Carleton University

October 30, 2021

CARLETON ECONOMICS WORKING PAPERS

Carleton UNIVERSITY

Department of Economics

1125 Colonel By Drive
Ottawa, Ontario, Canada
K1S 5B6

# Comparing Out-of-Sample Performance of Machine Learning Methods to Forecast U.S. GDP Growth

Ba Chu[*] Shafiullah Qureshi[†]

October 28, 2021

**Abstract**

We run a 'horse race' among popular forecasting methods, including machine learning (ML) and deep learning (DL) methods, employed to forecast U.S. GDP growth. Given the unstable nature of GDP growth data, we implement a recursive forecasting strategy to calculate the out-of-sample performance metrics of forecasts for multiple subperiods. We use three sets of predictors: a large set of 224 predictors [of U.S. GDP growth] taken from a large quarterly macroeconomic database (namely, FRED-QD), a small set of nine strong predictors selected from the large set, and another small set including these nine strong predictors together with a high-frequency business condition index. We then obtain the following three main findings: **(1)** when forecasting with a large number of predictors with mixed predictive power, density-based ML methods (such as bagging or boosting) can outperform sparsity-based methods (such as Lasso) for long-horizon forecast, but this is not necessarily the case for short-horizon forecast; **(2)** density-based ML methods tend to perform better with a large set of predictors than with a small subset of strong predictors; and **(3)** parsimonious models using a strong high-frequency predictor can outperform sophisticated ML and DL models using a large number of low-frequency predictors, highlighting the important role of predictors in economic forecasting. We also find that ensemble ML methods (which are the special cases of density-based ML methods) can outperform popular DL methods.

*AMS 2020 subject classifications:* 62P20, 68T07, 68T09, and 68T99.
*Keywords:* Lasso, Ridge Regression, Random Forest, Boosting Algorithms, Artificial Neural Networks, Dimensional Reduction Methods, MIDAS, and GDP growth.

[*]Department of Economics, Carleton University, 1125 Colonel By Dr., Ottawa, Ontario, Canada. Email: ba.chu@carleton.ca. Tel: +1 613-520-2600 (ext. 1546).

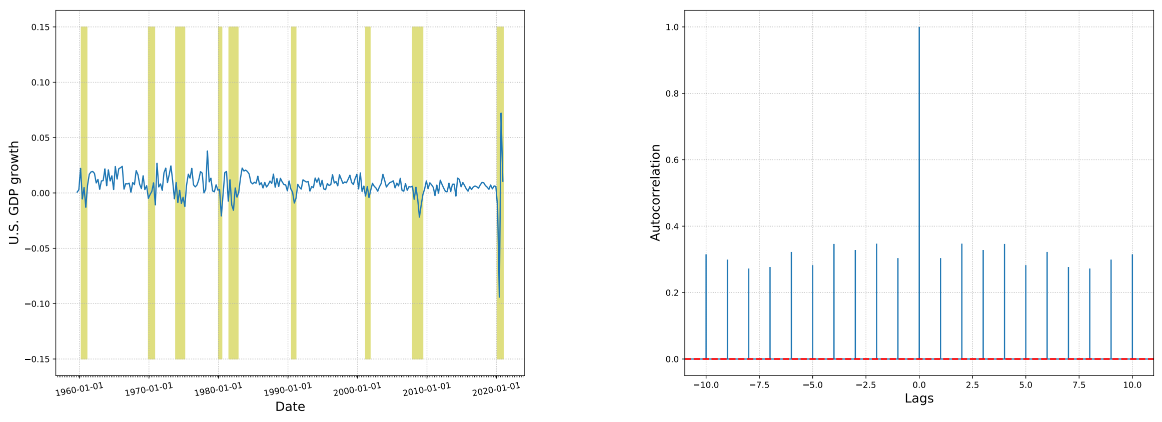
[†]Department of Economics, Carleton University, 1125 Colonel By Dr., Ottawa, Ontario, Canada. Email: shafiullah.qureshi@carleton.ca. Tel: +1 613-520-2600 (ext. 3778); and Department of Economics, NUML, Islamabad, Pakistan. Email: suqureshi@numl.edu.pk.

## 1 Introduction

Economic time series with low serial correlation and high volatility or structural changes are in general difficult to forecast accurately. Among the most important and interesting time series that are difficult to predict is the growth rate of GDP. Yet, GDP is one of the most useful measures to monitor macroeconomic developments. Since it often becomes available with time delay which hinders the effort to assess the current and future state of the economy, nowcasting and forecasting GDP growth are thus needed for an early assessment of the economic situation going forward.

The following graphs suggest that the U.S. GDP growth fluctuates considerably during the recession periods while the autocorrelations from lags one to ten are invariably below 0.4. In particular, the episode that started on 01/01/2020 when the GDP growth pulled back then surged sharply has never happened throughout the whole sample period. Therefore, in settings like this one involving a predictand with relatively low signal-to-noise ratio plus potential structural instabilities, evaluating and comparing the out-of-sample (OoS) performance of forecasting methods across different subperiods is a challenging, but important task.

Figure 1: Plots of U.S. GDP growth rates



(a) line plot (the shaded bars are the NBER recession dates)

(b) ACF plot

There is a large body of literature applying time series models and machine learning (ML) or deep learning (DL) methods to forecast GDP growth and other low-predictability macroeconomic variables.[1]

[1]DL methods are often considered as a subcategory of ML methods. In this paper, we refer to ML methods based on neural

Popular approaches include: dimensional reduction through common factors which are then used as additional predictors in a predictive regression model, or so-called factor-augmented regression (e.g., Stock and Watson, 2002a,b), bridge equations (e.g., Baffigi, Golinelli, and Parigi, 2004; Götz and Knetsch, 2019), univariate and multivariate mixed-frequency models using high-frequency variables to predict low-frequency variables (e.g., Foroni, Marcellino, and Schumacher, 2015; Ghysels, Santa-Clara, and Valkanov, 2004; McAlinn, 2021; Schorfheide and Song, 2015), machine learning (e.g., Giannone, Lenza, and Primiceri, 2021; Medeiros, Vasconcelos, Veiga, and Zilberman, 2021), and deep learning (e.g., Barkan, Benchimol, Caspi, Hammer, and Koenigstein, 2021; Nakamura, 2005; Paranhos, 2021). The literature has reached the following major conclusions: (1) factor-augmented models, time series models, and mixed-frequency models may not achieve any improvement in their forecast performance by using a large set of predictors (e.g., Carriero, Galvão, and Kapetanios, 2019). The reason may well be that either those models do not employ a shrinkage or variable selection device to prevent overfitting (which often occurs when there is a mix of many relevant and irrelevant predictors); thus the in-sample prediction accuracy cannot generalize well to unseen data, or there are only a few powerful predictors in the sample whereas the remaining predictors have little predictive power;[2] (2) when there is a large number of potentially useful predictors such that every predictor has some predictive information about the predictand, ML methods based on dense models which assume that all predictors are relevant (*or* density-based ML methods), such as boosting and bagging & bootstrapping, often produce better forecasts than sparse-modelling techniques which presume that the set of predictors is sparse, such as Lasso or subset selection (e.g., Giannone et al., 2021) – as argued by the authors, the superior performance of these density-based ML methods is ascribed to pervasive model uncertainty. Giannone et al. (2021) find evidence favouring dense models in several popular macroeconomic and financial datasets. However, when there is only a few strong predictors and the rest are irrelevant or have low signal-to-noise ratios, sparsification has often been advocated as a method to improve model interpretability and out-of-sample forecasts. For example, in a Bayesian time-varying parameter model, there is an empirical evidence that large sparse models can forecast really well (e.g., Huber, Koop, and Onorante, 2021 or Woody, Carvalho, and Murray, 2021). Indeed, Hastie,

networks as DL methods and those not based on neural networks as ML methods in general.

[2]Overfitting is referred to as the situation where the in-sample performance of a forecasting method is much better than its out-of-sample performance.

Tibshirani, and Friedman's (2009) Bet on Sparsity principle suggests that sparse-modelling methods can outperform density-based methods for high-dimensional problems if the bet is successful. Of course, Bet on Sparsity is not always guaranteed to succeed. Therefore, the debate over dense models or sparse models is not likely to end soon; (3) mixed-frequency models can perform well out-of-sample because they exploit information content of high-frequency variables to improve the accuracy of forecasts of a low-frequency variable; and (4) deep neural networks can provide better forecasts of highly volatile macroeconomic variables (such as inflation) than standard time series models, especially in the long term.

In light of the above discussion, we are interested in the following three main questions: (i) *how does the GDP growth forecast performance of dense-modelling ML methods compare with that of sparse-modelling ML methods and other time series models in a sample with a large number of predictors with mixed predictive power or in a sample with a small number of strong predictors?*, (ii) *which methods provide better forecasts with a large set of predictors than with a small subset of strong predictors?*, and (iii) *which methods can produce superior forecast performance overall across several large and small sets of predictors (to be defined below)?* This paper is an attempt to address those questions using a graphical approach instead of statistical inference approaches as most testing procedures for forecast comparison assumes that there is an underlying stationary, weakly dependent process generating data that an econometrician can recover by sampling from this process over a sufficiently long period of time. The above graph indeed suggests that the U.S. GDP growth may not behave like a stationary, weakly dependent process as the autocorrelation function decays very slowly at large lags, which thus makes it quite challenging to correctly rank forecast models with a statistical procedure. Testing methods, such as Giacomini and Rossi's (2010) Fluctuation and One-Time Reversal tests, can potentially be used, but the difficulty is to obtain correct critical values for these tests when the weak dependence assumption no longer holds.

We use a large quarterly macroeconomic database (namely, FRED-QD at the vintage date 2021-01) constructed by McCracken and Ng (2020) for the period from 1959Q1 to 2020Q4 with 225 variables and data on Aruoba, Diebold, and Scotti's (2009) business condition index (the ADS index) which tracks real-time business conditions for the U.S. economy.[3] We then use these datasets to form three sets of

[3]The ADS is constructed from six seasonally adjusted economic indicators with both high-frequency and low-frequency data. These indicators include: weekly initial unemployment claims, monthly payroll employment, monthly industrial production, monthly real personal income less transfer payments, monthly real manufacturing and trade sales, quarterly real GDP. The index is updated in real time as new data on the index's components is released.

predictors: the first set consists of 224 FRED-QD predictors, the second set consists of nine FRED-QD predictors [taken from the first set], which strongly comove with U.S. GDP growth, and the third set is the second set together with the ADS index.[4] For the first set of predictors, we implement seven ML and DL methods, including a method inducing sparsity (Lasso), a shrinkage-based method (ridge regression), three tree-based methods (XGBoost, Gradient Boosting Machine (GBM), and Random Forest), and two DL methods (deep neural networks (NN) and the long short-term memory model (LSTM)).[5] For the second set of predictors, we implement nine methods, including Lasso (which is equivalent to the ordinary least squares (OLS) as all predictors get selected in this case), ridge regression (Ridge), XGBoost, GBM, Random Forest, deep neural networks, LSTM, a new supervised principal component analysis (sPCA) which combines the conventional PCA with Yousuf and Feng's (2021) variable screening procedure based on the partial distance correlation, and a time series model with nonparametric trend and seasonal patterns (Facebook Prophet). For the third set of predictors, we also implement a mixed-frequency model in addition to the aforementioned nine methods. (A detailed description of all the forecasting methods is presented in Section 2.)

We then conduct real-time forecasting exercises. The timing of forecasts is as follows: once the GDP numbers up to time $t$ are released (in reality, the release may happen sometime after $t$ due to time delay), we produce forecasts for each period $t + h$ up to four quarters ahead ($h = 1, 2, 3, 4$ quarters). We only use the information available up to time $t$ to forecast for the period $t + h$, although we may have additional (monthly) information about the quarter $t + h$ at some point before time $t + h$.[6] We employ a rolling-window strategy to compute the OoS performance metrics of forecasts for each horizon, $h$ : the root mean squared error (RMSE), the mean absolute error (MAE), and the OoS $R^2$ for over 145 rolling blocks of observations, or sub-samples, taken from the whole sample period. The first sub-sample runs from 9/1/1959 to 6/1/1984, and the next sub-sample is formed by shifting the last sub-sample forward

[4]Note that, since the observations on the ADS index are available at a higher frequency than those on U.S. GDP growth, this set of predictors is constructed by adding to the second set of predictors five (monthly) observations on the ADS index available at/before each point in time when one starts forecasting the future values of GDP growth. This sample construction, which is reminiscent of the unrestricted mixed data sampling proposed by Foroni et al. (2015), allows us to implement various ML and DL methods using a dataset with variables of different sampling frequencies.

[5]The "deep" in deep NN simply refers to the multiple layers of the neural network whereas a shallow network often has a single layer.

[6]If information in the quarter $t + h$ (such as the ADS indices of the first/second month of this quarter) is employed to improve a forecast, this forecast then becomes a nowcast instead.

by one quarter. We then use the obtained sequence of over 145 values of an OoS performance metric to evaluate the forecast performance of each method. We also compare the forecast performance across methods and sets of predictors. (We shall give details about this recursive forecasting strategy in Section 3.) This strategy is basically an acknowledgement that a predictive relationship may undergo structural changes over time [perhaps due to the nonstationarity of the underlying data generating process]. Thus, our rolling-window estimation approaches using continuously updated sub-samples is an ad hoc method to address the structural change issue in ML forecasting models.

The results of our forecasting experiments suggest the following answers to the above three questions. Regarding Question (i), using the first set of predictors, there is no clear-cut advantage of density-based ML methods over sparsity-based ML methods for one-period forecast while density-based ML methods can outperform sparsity-based ML methods for long-horizon (i.e., $h > 1$) forecast. Using the second set of predictors, the Lasso OLS method can outperform density-based ML methods for one-period ahead forecast. In contrast, density-based ML methods (i.e., Random Forest) or the sPCA can produce better longer-horizon forecasts. When a strong high-frequency predictor is available, a mixed-frequency model can generate the best long-horizon forecasts. At the same time, sPCA or Lasso can produce better one-period ahead forecasts than a mixed-frequency method. Regarding Question (ii), the short answer is that Random Forest, XGBoost, and GBM can potentially perform better with a large set of predictors than with a small set of strong predictors for short-to-medium horizon forecast; however, there is not much difference in their performance for long-horizon forecast. Deep NN performs better with a small set of strong predictors than with a large set of predictors only for long-horizon forecast while it seems to perform better with a large set of predictors for one-period ahead forecast. Ridge regression always performs better with a small set of strong predictors than with a large set of predictors. These findings corroborate Giannone et al.'s (2021) suggestion that ensemble methods, such as boosting and random forest, can benefit from the availability of many predictors while this is not necessarily the case for other methods, such as ridge regression or DL. Regarding Question (iii), sPCA (which uses only a small set of strong predictors together with the ADS index) can provide better OoS performance for short-horizon forecast. Meanwhile, a mixed-frequency model can be the winner for long-horizon forecast. The reason for this behaviour is that overfitting is a critical issue in machine learning and the risk of overfitting increases with the forecast horizon (because the pattern of data to be predicted becomes more different from the pattern of data observed as the forecast horizon increases). Also, when data have a low signal-to-noise ratio, there is no priori reason to call for any particular type of nonlinearity. Therefore, parsimonious models with a good high-frequency predictor can outperform more sophisticated nonlinear ML and DL models.

We thus make several contributions. First, we conduct an extensive OoS forecast experiment by comparing the forecast performance of many popular methods over time and across various sets of predictors. Many existing works only perform OoS forecast comparisons based on an entire sample of observations – a model is estimated using an initial training sample, then it is recursively re-estimated by rolling/expanding this training sample until the end of the whole sample period in order to compute forecasts for a specific number of periods ahead; the value of an OoS performance metric can thus be calculated for these forecasts. However, in practice, the evaluation of the OoS forecast performance (over time) of methods is often required in order to conduct a fair comparison of these methods, especially when the underlying variables display episodes of abrupt changes or low/high fluctuation. Therefore, a good forecasting method should ideally perform well across different subperiods. Second, we find that density-based ML methods can outperform sparsity-based ML methods for long-horizon forecast, but this is not necessarily the case for short-horizon forecast. A parsimonious forecasting model may outperform more sophisticated models when there is a high-frequency predictor with superior predictive power, highlighting the vital role of predictors in economic forecast. When there is a large number of predictors (not necessarily with equal predictive power), ensemble methods (such as Random Forest) can then deliver decent forecast performance in many cases. Finally, we also find that ensemble methods (such as boosting or bagging) can outperform many popular DL methods in this context of GDP growth forecasting. This finding is new relative to the existing literature which demonstrates the superior forecast performance of both ML and DL methods and suggests no particular ranking of those methods in the context of equity risk premium forecast or bond excess return forecast (see, e.g., Bianchi, Büchner, and Tamoni, 2021; Gu, Kelly, and Xiu, 2020). From a theoretical perspective, there has not been any proof that DL methods outperform ensemble ML methods or vice versa in supervised learning problems (Athey and Imbens, 2019).

The rest of this paper is organized as follows. Section 2 provides a brief description of the forecasting methods employed in this paper. Although this description may appear non-technical and thus primarily targets readers with modest background in ML methods, interested readers can find rigorous treatment of these algorithms in the original papers or in many good ML textbooks, for example, Hastie et al. (2009) and Murphy (2012). Our proposed forecast evaluation strategy is explained in Section 3. Section 4 presents a detailed description of the data and the main results of the paper. Section 5 gives some concluding remarks.

## 2 Description of Forecasting Methods

### 2.1 Mixed Data Sampling (MIDAS)

When one forecasts a low-frequency variable (e.g., the GDP growth in our case) using a high-frequency covariate (such as the ADS index), observations of the latter need to be aggregated using some weighting scheme to match the sampling rate of data on the low-frequency variable. MIDAS employs distributed lag polynomials (often parametrized by a few parameters to avoid parameter proliferation) as weighting functions. Let $Y_t$ represent a low-frequency variable, say quarterly GDP growth, and let $X_t$ be a high-frequency variable, say a monthly economic index. Past observations of $Y_t$ can be retrieved by the low-frequency (LF) lag operator $L_{LF}$ (i.e., $L^k_{LF}Y_t = Y_{t-k}$ is the observation which is $k$ *quarters* before the start time $t$ of a quarter). Similarly, past observations of $X_t$ can be retrieved by the high-frequency (HF) lag operator $L_{HF}$ (i.e., $L^k_{HF}X_t$ is the observation of $X$ which is $k$ *months* before $t$). To compute the $h$-period ahead forecast $\widehat{Y}_{t+h}$ associated with $Y_{t+h}$, we employ the following MIDAS model proposed by Ghysels, Sinko, and Valkanov (2007):

$$Y_{t+h} = \beta_0 + \sum_{i=0}^{p} \beta_i L^i_{LF} Y_t + \gamma \sum_{k=1}^{m} \Phi\left(k; \boldsymbol{\theta}\right) L^k_{HF} X_t + \epsilon_{t+h}, \tag{2.1}$$

where $\epsilon_{t+h}$, $t = 1, \ldots, T$, are i.i.d. errors; $m = 3$ because a quarter has three months; and $\Phi\left(k; \boldsymbol{\theta}\right)$ is a distributed lag polynomial, for example, it can have an exponential Almon specification: $\Phi\left(k; \theta_1, \theta_2\right) := \frac{\exp\left(\theta_1 k + \theta_2 k^2\right)}{\sum_{j=1}^{m} \exp(\theta_1 j + \theta_2 j^2)}$ or a beta specification as suggested by Ghysels et al. (2004).

## 2.2 Penalized Regression

The least-squares estimator often produces estimates with low bias, but the variances of these estimates and of the predicted values can increase with the number of covariates in a linear regression model. These variances can grow to infinity when the number of covariates is greater than the sample size. Therefore, the only way to increase the prediction accuracy while decreasing overfitting so that the estimated model can better predict unseen data is by reducing the variances of the estimates. This reduction can be achieved by shrinking slope coefficients to zero. Ridge regression proposed by Hoerl and Kennard (1970) shrinks all the regression coefficients to small numbers close to zero. In a predictive regression model, $Y_{t+h} = \beta_0 + \sum_{i=1}^{p} \beta_i X_{i,t} + \epsilon_{t+h}$, the ridge estimates of the coefficients minimize the following penalized sum of squared errors:

$$\widehat{\boldsymbol{\beta}}^{(ridge)} := \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{t=1}^{T} \left( Y_{t+h} - \beta_0 - \sum_{i=1}^{p} \beta_i X_{i,t} \right)^2 + \lambda \sum_{i=1}^{p} \beta_i^2 \right\},$$

where $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage. In a typical forecasting exercise, a data-driven value of $\lambda$ should be adaptively chosen via cross validation to minimize the OoS mean squared forecast error.

On the other hand, Lasso introduced by Tibshirani (1996) shrinks the coefficients of irrelevant covariates to exactly zeros. Therefore, the Lasso estimator defined by the following minimization problem yields a sparse solution:

$$\widehat{\boldsymbol{\beta}}^{(lasso)} := \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{t=1}^{T} \left( Y_{t+h} - \beta_0 - \sum_{i=1}^{p} \beta_i X_{i,t} \right)^2 + \lambda \sum_{i=1}^{p} |\beta_i| \right\}.$$

Because the ridge and Lasso regressions use different types of regularization, the ridge regression is mainly used to handle the non-invertibility problem of the least squares estimator when there are many covariates which are possibly multicollinear or when there are more covariates than observations while the Lasso regression can also be employed as a device for variable selection (apart from being a method to handle the non-invertibility problem).

## 2.3 Supervised Principal Component Analysis (sPCA)

As discussed above, forecasting with standard multivariate regression using many covariates suffers from poor in-sample and out-of-sample performance due to 'the curse of dimensionality'. The PCA can reduce a large number of predictors to a few principal components (which are just linear combinations of those predictors), and these components merely summarize most comovement within the predictors (but they do not incorporate any variability in the mean of each predictor). The PCA also ignores the forecasted variable, thus it is an unsupervised ML technique. If there are some irrelevant predictors with low signal-to-noise ratios, the obtained principal components can potentially be contaminated by these predictors. Therefore, forecasts generated using these principal components as covariates in a low-dimensional predictive regression model tend to have inferior performance. The sPCA proposed by Bair, Hastie, Paul, and Tibshirani (2006) overcomes this drawback of the PCA by first screening out irrelevant predictors, and then constructing the principal components out of the predictors with the strongest correlation with the forecasted variable. These [supervised] principal components will be used as covariates in a predictive regression to make forecasts as the final step. An important advantage of the sPCA over the partial least squares (PLS) method is its use of thresholding to remove all the predictors with low correlations with the target variable while the PLS retains all predictors, thus can be affected by the noise in the irrelevant predictors.

In this paper, we improve the performance of the sPCA by replacing its variable screening step with a powerful procedure based on the partial distance correlation proposed by Yousuf and Feng (2021). We have observed a significant improvement in the quality of out-of-sample forecasts as the result of applying this procedure. Suppose that the variable screening procedure finds $N$ strongest predictors. We then proceed to apply the standard PCA to construct principal components for these predictors. We can formulate a factor-augmented predictive regression model as follows: Given a balanced panel of $N$ standardized series, $\boldsymbol{X}_t := (X_{1,t}, \ldots, X_{N,t})$ for $t = 1, \ldots, T_1 \ll T$, as a training sample, the PCA summarizes the sample covariability of these series in this panel via $r$ factors, $f_{1,t}, \ldots, f_{r,t}$ with $r$ vectors of loadings, $\boldsymbol{\lambda}_i := (\lambda_{i,1}, \ldots, \lambda_{i,N})$ for $i = 1, \ldots, r$, where $(\gamma_i, \boldsymbol{\lambda}_i)$ is the $i$-th eigenvalue-eigenvector pair of the $N \times N$ sample variance-covariance matrix of $\boldsymbol{X}_t$ with $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_N \geq 0$.[7] Each factor is thus a linear

[7]In view of McCracken and Ng (2020), to form the panel dataset $\boldsymbol{X}_t$, $t = 1, \ldots, T_1$, we replace any missing value in a series

combination of $\boldsymbol{X}_t$, say $f_{i,t} := \sum_{n=1}^{N} \lambda_{i,n} X_{n,t}$, $i = 1, \ldots, r$. Therefore, the total variation in those $N$ series can be most accounted for by these $r$ factors. Forecasts can then be computed by the projection onto the linear space of the obtained common factors:

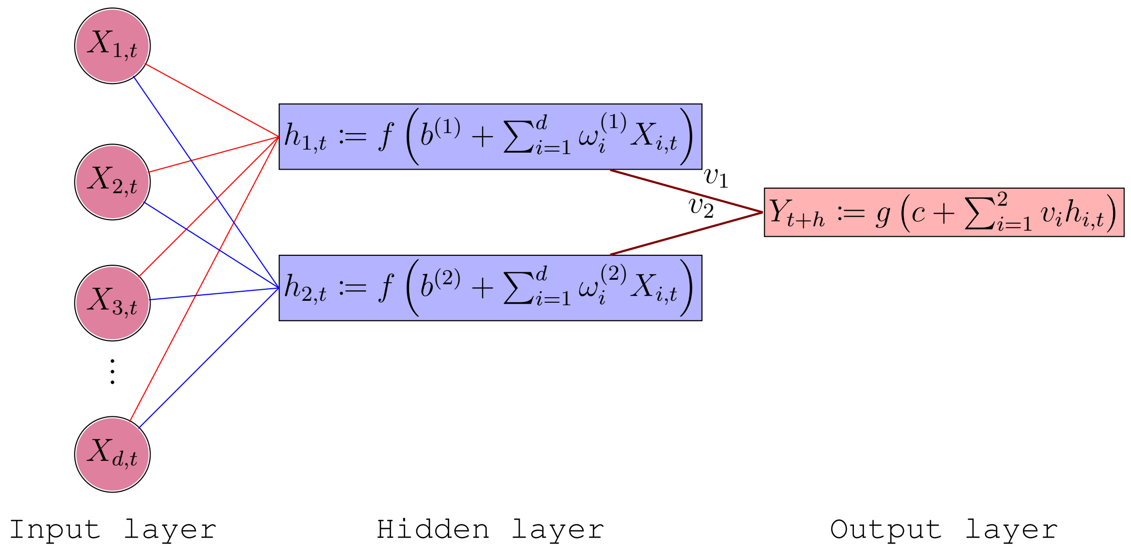
$$Y_{t+h} = \beta_0 + \beta_1 f_{1,t} + \cdots + \beta_r f_{r,t} + \epsilon_{t+h}, \tag{2.2}$$

where the number of factors $r$ needs to be determined beforehand via an information criterion or a scree plot. McCracken and Ng (2020) suggest that $r = 7$. It is important to note at this point that the eigenvectors $\boldsymbol{\lambda}_i$ appearing in the above expression of the $i$-th factor $f_{i,t}$ must be computed using only the training sample to avoid look-ahead biases. Given these values of $\boldsymbol{\lambda}_i$, rolling-window forecasts can be constructed by using the standard procedure applied to the linear model (2.2) above, because each factor, $f_{i,t}$, can always be calculated from the standardized series $\boldsymbol{X}_t$, $t = T_1 + 1, \ldots, T$ in the testing sample.

## 2.4 Deep Learning

The standard framework for deep learning is based on artificial neural networks (ANNs). An ANN describes the relationship between input signals and output signals by using a model of interactions between neurons inspired by the structure of the biological brain. Signals received by neurons at the input level are weighted according to their relative importance before being aggregated, transformed, and passed to neurons at the next hidden level. The whole process is then repeated until all output signals generated from neurons at the hidden level are received by the neuron at the output level. A simple layout of an ANN is shown in the following diagram, where there are $d$ neurons, $X_{i,t}$, $i = 1, \ldots, d$, at the input layer, representing $d$ covariates at time $t$. These covariates are then combined using two different sets of weights, say $b^{(j)} + \sum_{i=1}^{d} \omega_i^{(j)} X_{i,t}$ for $j = 1, 2$. These weighted combinations are then transformed into output signals via an activation function $f(\cdot)$, before being passed to two different neurons, say $h_{1,t}$ and $h_{2,t}$, at the hidden layer. The signals from these neurons are then weighted with weights, $v_1$ and $v_2$, and transformed one more time, using a possibly different activation function, $g(\cdot)$, before being passed onto the final neuron at the output layer.

by the unconditional sample mean, then standardize each series so that it has zero mean and unit variance. Note that we have calculated the time-series means and variances by using only the observations $t = 1, \ldots, T_1$ to guard against look-ahead biases.



$X_{1,t}$
$X_{2,t}$
$X_{3,t}$
$\vdots$
$X_{d,t}$
$h_{1,t} := f\left(b^{(1)} + \sum_{i=1}^{d} \omega_i^{(1)} X_{i,t}\right)$
$h_{2,t} := f\left(b^{(2)} + \sum_{i=1}^{d} \omega_i^{(2)} X_{i,t}\right)$
$v_1$
$v_2$
$Y_{t+h} := g\left(c + \sum_{i=1}^{2} v_i h_{i,t}\right)$
Input layer
Hidden layer
Output layer

Activation functions play a major role in transforming input signals from neurons at each layer of an ANN to output signals, which will then be propagated to other neurons at the next layer. This transformation is meant to capture nonlinearities in the relationships between input and output signals. Some popular activation functions include: the sigmoid function (a so-called logistic function) mapping a real number to the interval $(0, 1]$, the tanh function mapping a real number to the interval $[-1, 1]$, the rectified linear unit (ReLU) function truncating a real number, $x$, below a threshold of zero: $\max(0, x)$, and a linear function. The sigmoid and tanh functions are mainly used for classification problems while others can be used for regression problems. In this illustration, the learning process involves finding the biases: $b^{(j)}$ for $j = 1, 2$ and $c$, and the weights: $\omega_i^{(j)}$, $i = 1, \dots, d$, and $v^{(j)}$ for $j = 1, 2$ to minimize a loss function of forecast errors, $\epsilon_{t+h} := Y_{t+h} - g\left(c + \sum_{i=1}^{2} v_i h_{i,t}\right)$, $t = 1, \dots, T$. These errors can be represented in terms of observations, $X_{i,t}$, $i = 1, \dots, d$, by back-propagating the signals $h_{1,t}$ and $h_{2,t}$ from the hidden layer all the way back to the neurons at the input layer. In practice, an ANN can have many hidden layers; thus there are a large number of parameters to be learnt and evaluating the errors involves multiple stages of back-propagation. Stochastic gradient algorithms with adaptive learning rates are often employed to minimize this loss function. One of the most successful algorithms as such is the adaptive moment estimation (Adam) proposed by Kingma and Ba (2017), which will be employed to train our DL models.

As seen in the block diagram above, an ANN is a static neural network in the sense that the output signals at the hidden layer are only driven by the input signals at the input layer contemporaneously. Therefore, these output signals do not memorize their values nor the values of the input signals at any previous point in time. Recurrent neural networks (RNNs) were introduced to remedy this null-memory problem of ANNs. To allow for some memory in the output signals $h_{1,t}$ and $h_{2,t}$, one can use a simple intertemporal formulation, such as:

$$h_{j,t} = f\left(b^{(j)} + \sum_{i=1}^{2} \alpha_i h_{i,t-1} + \sum_{i=1}^{d} \omega_i^{(1)} X_{i,t}\right) \text{ for } j = 1, 2, \tag{2.3}$$

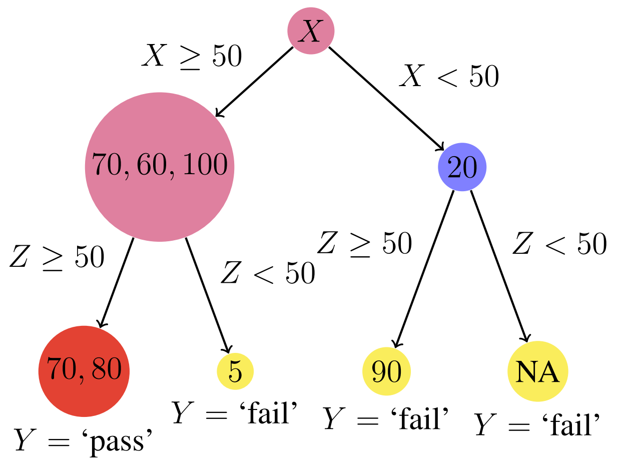
where $\alpha_1$ and $\alpha_2$ are the time-invariant weights of the output signals at the hidden layer at time $t - 1$. Therefore, a typical RNN has more parameters than a typical ANN. To estimate those parameters, one again needs to minimize a loss function evaluated on the errors $\epsilon_{t+h}$, $t = 1, \ldots, T$, which can be represented in terms of $X_{i,t}$, $i = 1, \ldots, d$, and $(h_{1,0}, h_{2,0})$ by recursively backward-substituting the latent variables $h_{1,t}$ and $h_{2,t}$ to the initial time $t = 0$. Since Model (2.3) behaves like a nonlinear autoregressive process with exogenous regressors, the value of $h_{j,T}$ for some large $T$ can explode or vanish, depending on whether the values of $\alpha_1$ and $\alpha_2$ are large or small respectively. Hence, the gradients of the loss function with respect to these time-invariant weights can also explode or vanish [see, e.g., Bengio, Frasconi, and Simard (1993)]. To alleviate the problem of vanishing or exploding gradients, Hochreiter and Schmidhuber (1997) proposed the long short-term memory model (LSTM) as an improved version of RNNs. The mathematical formation of this model is quite complicated to explain in simple terms. The basic idea is that the LSTM includes other mechanisms (namely, the forget gate, the external input gate, and the output gate) to remember or forget input and output signals from neurons pertaining to a particular input/hidden layer at several previous points in time so that derivatives of the loss function do not vanish nor explode.[8]

The performance of a DL model depends on several characteristics of the neural network and the optimizer used to minimize the loss function. Such characteristics are called the hyperparameters of this model. These hyperparameters include the number of hidden layers, the number of neurons in each hidden layer, the activation function, the dropout rate, the learning rate, and so on. Fine-tuning a model is to determine an optimal combination of the hyperparameter values so that the out-of-sample performance of the fine-tuned model becomes as close as possible to its in-sample performance.

[8]Mathematical equations of these gates can be found, for example, in Goodfellow, Bengio, and Courville (2016, chapter 10).

## 2.5 Random Forest

The random forest (RF) algorithm for classification and regression was developed by Ho (1995) and Breiman (2001). This algorithm combines many decision/regression trees grown independently using sub-samples drawn from the original sample. This combination often leads to a significant reduction in the mean squared forecast error compared with the case where an individual tree is being used. Trees are the building blocks of the RF. To give an example of decision trees, let's take a dataset that has a binary outcome variable ($Y$ = 'pass', 'fail', 'pass', 'fail') related to two covariates: mid-term grade and final grade ($X = 70, 20, 60, 100$ and $Z = 70, 90, 80, 5$ respectively). Assume that we have a rule – an outcome of 'pass' is assigned when both $X$ and $Z$ are above 50, otherwise an outcome of 'fail' is assigned. We can immediately construct a decision tree as in the following diagram:



In the above diagram, data on the covariates are sequentially divided into partitions (i.e., the observations of $X$ are split into two subsets, and given each of these subsets the observations of $Z$ are then split accordingly). At the end of this process, the leaf nodes are labelled according to the values of $Y$. In this example, the splitting rule (defined by the threshold value of $50$) used to grow the decision tree is given. Thus, one can predict any value of the outcome $Y$ for given values of $X$ and $Z$ by simply applying this rule. If this rule is not given, one can effectively learn the optimal rule from data available by growing many different decision trees corresponding to various rules, and then select the rule which produces partitions as different from each other as possible (and the numbers in each partition are as similar to each other as possible, and thus have similar outcomes). The rule satisfying this criterion is said to minimize the impurity measure. The idea of regression trees is pretty similar to that of decision trees if we can imagine that $Y$ can take many different values on the real line instead of binary values while the observations of X and Z can be split into multiple subsets instead of two subsets as in this illustration. In this case, there will be much more leaf nodes (each of them corresponding to a bin of values of $Y$ on the real line).

A tree is a weak learner as in many cases it can only predict the outcome variable slightly better than a random number due to its high variance. By the central limit theorem, averaging many independent weak learners can reduce the variance, and thus make it less susceptible to overfitting – which is the main motivation behind the RF. Therefore, the baseline RF algorithm can be implemented in the following four steps:

1. Draw a bootstrap sample of size less than or equal to the original sample size.
2. Grow a decision tree from each bootstrap sample. At each node,
    a. Randomly select a subset of features without replacement.
    b. Split the node using the feature that provides the best split using the threshold obtained by minimizing the Gini impurity or the cross entropy (as described in the above tree diagram).
3. Repeat the above two steps $k$ times to create $k$ independent trees.
4. Aggregate the predictions produced by those $k$ independent trees.

For time series data [which have a natural ordering], the step 1 of the above RF algorithm is usually implemented via a "block bootstrap" method according to which a sufficiently long block of time-series observations is resampled in order to capture serial dependence in the block. The *regular block bootstrap* does this with a fixed block length while the *stationary bootstrap* uses random block lengths, where the length may be drawn from an auxiliary distribution (see, e.g., Lahiri (2003) for a detailed discussion of dependent bootstrap methods). Decreasing the size of the bootstrap sample improves the diversity of trees as the probability that a particular observation is included in all the bootstrap samples is lower. This diversity may yield a more random forest, which can help to reduce the effect of overfitting. On the contrary, increasing the size of the bootstrap sample can exacerbate the overfitting problem as trees are more likely to become similar to each other, and therefore learn to fit the original training data more closely. However, an issue with block bootstrap is that points around each joint of two adjacent blocks may not well mimic the serial dependence structure in the original sample. Therefore, if the time series is long, one could implement sub-sampling (i.e., sampling just one block at a time) instead. Another potential sampling scheme that could be used in the first step of the RF algorithm is Dahl and Sørensen's (2021) resampling using generative adversarial networks. This method can effectively generate many time series that mimic the serial dependence structure of the original series.

## 2.6 Gradient Boosting

Gradient boosting is one of the most successful ML algorithms. Unlike Random Forest which utilizes the *bootstrapping and bagging* principle (i.e., each tree is built on a sub-sample randomly drawn from the entire dataset, and many independent trees will then be aggregated to form a random forest), Gradient Boosting does not use bootstrap. Instead, it boosts weak learners (typically, using the same decision/regression trees as Random Forest) iteratively by focussing more on observations that are difficult to predict in the previous iterations, and then combine those weak learners with more weights given to better learners (that predict the outcome variable with a better accuracy). Given an outcome variable, $Y$, and a covariate, $X$, the gradient boosting predictive regression can be implemented as in the following steps:

1. Fit a regression tree, $F_1(x)$, to the target $Y_{t+h}$ using the feature $X_t$, then compute the error $\epsilon_{1,t+h} := Y_{t+h} - F_1(X_t)$. This error captures any variation in $Y_{t+h}$ not captured by the regression tree $F_1$.

2. Fit another regression tree, $F_2(x)$, to the target $\epsilon_{1,t+h}$ using the same feature $X_t$, then calculate the error $\epsilon_{2,t+h} := \epsilon_{1,t+h} - F_2(X_t)$. This error captures any variation in $Y_{t+h}$ not captured by both the trees $F_1$ and $F_2$.

3. We continue to apply the above procedure until the iteration $n-1$, where we have an error, $\epsilon_{n-1,t+h}$. Fitting the $n$-th regression tree, $F_n(x)$, to $\epsilon_{n-1,t+h}$ with the covariate $X_t$ yields the $n$-th error $\epsilon_{n,t+h} = \epsilon_{n-1,t+h} - F_n(X_t)$.

4. Given $n$ regression trees fitted in the above steps, we score them in terms of their prediction accuracy with scoring weights, $\gamma_i$ for $i = 1, \ldots, n$. The final boosting predictive regression model can then be constructed as $Y_{t+h} = \sum_{i=1}^{n} \gamma_i F_i(X_t)$.

XGBoost (eXtreme Gradient Boosting) is a new boosting algorithm introduced by Chen and Guestrin (2016). XGBoost has proven its superior performance in many applications due to its high predictive accuracy and lighting speed. XGBoost is different from the gradient boosting described above in the following two important aspects: (a) it can penalize the complexity of regression trees and (b) it randomly samples a subset of covariates in each boosting step. The purpose of introducing these two regularization addons is to reduce overfitting so that a trained XGBoost model can better predict unseen data. The cost of this generalization is that XGBoost has many more hyperparameters to tune than the RF or the standard gradient boosting. Therefore, validation of XGBoost models often consumes more time and requires more domain expertise from users to achieve good prediction accuracy.

## 2.7 Facebook Prophet

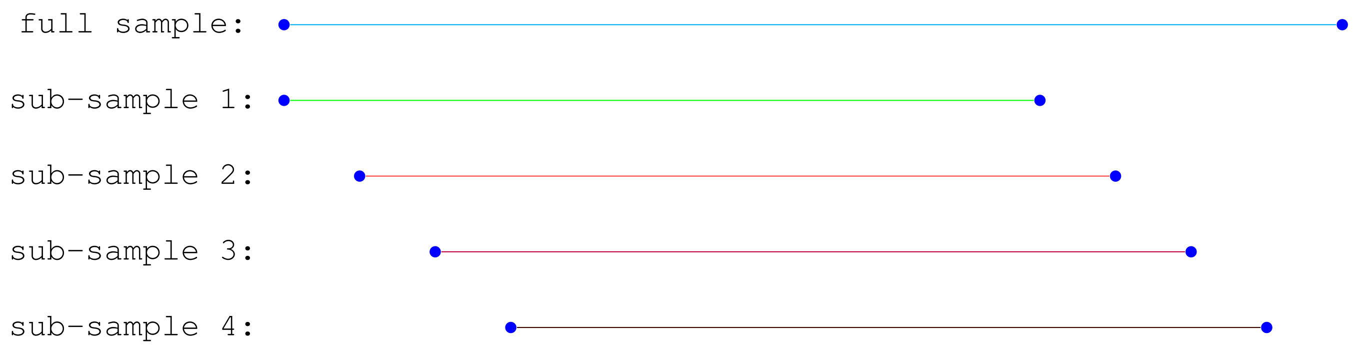
Prophet was developed at Facebook by Taylor and Letham (2018) as a flexible and highly customable tool for business and economic forecasting. Many conventional forecasting procedures (such as ARIMA or the exponential smoothing) fail to capture nonlinear trend or seasonal patterns in data, which are often the main catalysts for poor forecasts produced by these procedures. Prophet can handle any of the following attributes in time series data: (a) strong seasonal patterns occurring daily, weekly, monthly, and/or yearly, (b) nonlinear trends, and (c) holidays and other one-time events that occur irregularly. Therefore, as Taylor and Letham (2018) suggest, Prophet can generally yield forecasts which are as good as other nonlinear time series models with a fractional amount of time and effort. A predictive Prophet model can then be written as an additive time series regression model:

$$Y_{t+h} = g(t+h) + s(t+h) + v(t+h) + \boldsymbol{X}_t^\top \boldsymbol{\beta} + \epsilon_{t+h}, \tag{2.4}$$

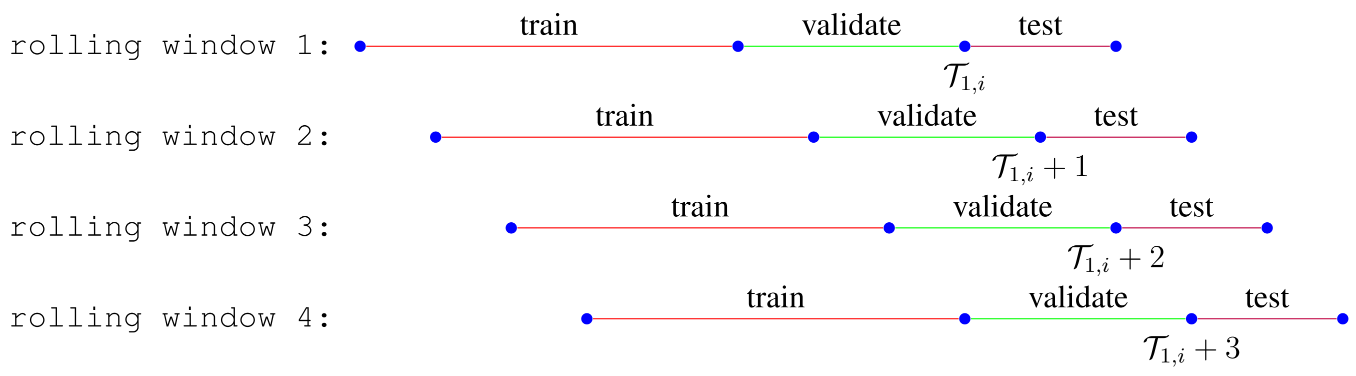
where $g(t)$ is a trend function that can be modelled as a piecewise linear function with multiple breakpoints, $s(t)$ captures seasonal effects by using standard Fourier series, $v(t)$ is a linear function of dummy variables of points in time when there are holidays or one-time events occurring, $\boldsymbol{X}_t$ is a vector of predictors, and $\epsilon_{t+h}$ is an i.i.d. error term. This model is estimated by a Bayesian approach, thus confidence intervals of forecasts can be immediately obtained.

## 3 Recursive Forecasting Strategy

To evaluate the OoS performance of a forecasting model over time, we construct many rolling sub-samples from a given sample, then compute the OoS performance metrics (to be defined below) for each of these sub-samples. In the following diagram, `sub-sample 2` is constructed by removing the first observation from `sub-sample 1` and extending forward one more observation. Similarly, `sub-sample 3` is created by removing the first observation from `sub-sample 2` and extending forward one more observation.



To compute the performance metrics of $h$-period ahead forecasts for each sub-sample, $i = 1, \dots, N$, we employ the rolling-window strategy where multiple rolling windows of observations are constructed from this sub-sample. For example, in the following diagram, `rolling window 2` is formed by moving `rolling window 1` forward one observation, `rolling window 3` is formed by moving `rolling window 2` forward one observation, and so on. Observations in each window is then split into three parts used to train, validate, and test a model. Note that we set the same sizes for training, validating, and testing samples in all rolling windows (for example, we used 45 observations for training a model, 15 observations for validating this model, and the last four observations used to compute one- to four-period ahead forecast errors for the validated model).



We take the following steps to produce a $h$-period ahead forecast in each rolling window:

*Step 1:* Estimate a model with the training sample, and at the same time, use the validating sample to fine-tune this model so that the RMSE of OoS forecasts [of validation data] is minimal (i.e., to choose the optimal hyperparameter values for the model so that the effect of overfitting can be reduced, and thus the OoS forecast performance can be improved).[9]

*Step 2:* Use the fine-tuned model obtained in *Step 1* to produce a $h$-period ahead forecast. The actual observation corresponding to this forecast in the test sample can then be used to compute the forecast error.

Let $\widehat{Y}_{i,t+h}$, $t = \mathcal{T}_{1,i}, \ldots, \mathcal{T}_i - h$, represent the [$h$-period ahead] forecasts associated with the actual observations $Y_{i,t+h}$, $t = \mathcal{T}_{1,i}, \ldots, \mathcal{T}_i - h$, in sub-sample $i$, where $\mathcal{T}_i$ is the size of this sub-sample. In this case, we have $\mathcal{T}_i - h - \mathcal{T}_{1,i} + 1$ rolling windows. We evaluate these forecasts with three performance metrics: the root mean squared error (RMSE), the mean absolute error (MAE), and the OoS $R^2$ :

$$RMSE_i = \sqrt{\frac{1}{\mathcal{T}_i - h - \mathcal{T}_{1,i} + 1} \sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} e_{i,t+h}^2}, \tag{3.1}$$

$$MAE_i = \frac{1}{\mathcal{T}_i - h - \mathcal{T}_{1,i} + 1} \sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} |e_{i,t+h}|, \tag{3.2}$$

[9]Optimal hyperparameters (for instance, in DL, they include the dropout rate, the learning rate, the number of neurons in each layer, and the number of hidden layers) are often selected using a popular data-driven method called cross validation (CV). In CV, the joint domain of hyperparameters is divided into grid points (and each grid point is associated with a combination of hyperparameter values). The model is estimated using only the training data for each combination of hyperparameter values (corresponding to each grid point). This estimated model is then used to make predictions on the validating sample. The forecast errors on this validating sample are used to compute the RMSE. The optimal hyperparameter values are the values with the minimum RMSE value. Therefore, the main idea behind CV is to check the actual forecast performance of a method using a dataset that is different from the one used to train the model.

$$R_i^2 = 1 - \frac{\sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} e_{i,t+h}^2}{\sum_{t=\mathcal{T}_{1,i}}^{\mathcal{T}_i - h} \left(Y_{i,t+h} - \overline{Y}_{i,t+h}\right)^2}, \tag{3.3}$$

where $e_{i,t+h} := Y_{i,t+h} - \widehat{Y}_{i,t+h}$ is the forecast error at time $t+h$ in sub-sample $i$, and $\overline{Y}_{i,t+h}$ is a baseline forecast (which is the sample mean of observations from the beginning of the first window till time $t$ in sub-sample $i$). The OoS $R^2$ takes value in between $-\infty$ and one. Therefore, a negative value of the OoS $R^2$ means that the baseline forecast is better (in terms of the RMSE) than a forecast model while a positive value merely means that the forecast model is better than the baseline forecast. As the value of the OoS $R^2$ is closer to one, the forecast becomes better than the baseline forecast.

## 4 Results

### 4.1 Data Description

We use a large quarterly macroeconomic database (namely, FRED-QD) constructed by McCracken and Ng (2020) for the period from 1959Q1 to 2020Q4. This dataset consists of 248 macroeconomic and financial variables categorized into 14 groups: (1) National Income and Product Accounts, (2) Industrial Production, (3) Employment and Unemployment, (4) Housing, (5) Inventories, Orders, and Sales, (6) Prices, (7) Earnings and Productivity, (8) Interest Rates, (9) Money and Credit, (10) Household Balance Sheets, (11) Exchange Rates, (12) Consumer Sentiment and Uncertainty Index, (13) Stock Markets, and (14) Non-Household Balance Sheets. (A detailed list of variables in each of these groups is given in the technical appendix of McCracken and Ng (2020).) We also stabilize all the variables (so that they behave more like stationary time series) by using the same data transformations suggested by McCracken and Ng (2020). We have also removed 23 variables (as listed in Table 2) because they contain more than 50 missing observations at the beginning of the sample, and thus there are 225 variables remained for the analysis.

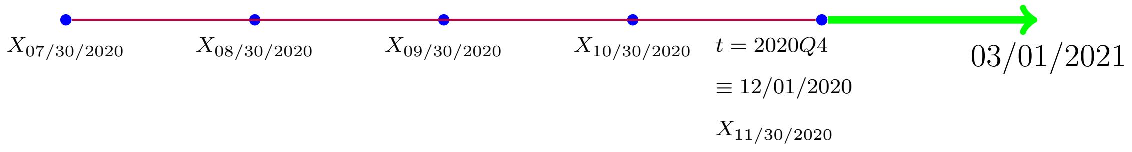
We forecast U.S. GDP growth using one of the following three sets of predictors (mentioned in the Introduction):

Preds (i) : 224 FRED-QD predictors,

Preds (ii) : Nine FRED-QD predictors (as listed in Table 1) selected from the 224 variables in Preds (i) above by Yousuf and Feng's (2021) variable screening procedure based on the partial distance correlation,

Preds (iii) : Preds (ii) plus the ADS index available at the daily frequency. (Figure 2 suggests that all these variables highly co-move with the GDP growth.)

Since it is not straightforward to apply ML and DL methods to a sample where the difference in sampling frequencies between the predictand and the predictors is large, we convert daily data on the ADS index to monthly data by taking the sample mean of daily observations in each month. Then, at each point in time when the forecast for a future quarterly GDP growth rate is constructed, we use five (monthly) observations on the ADS index at/before this point in time as predictive information to be fed into ML algorithms. For example, as illustrated in the following timeline, we are currently at time $t = 12/01/2020$ of 2020Q4 and we want to forecast GDP growth for 2021Q1 (ending on 03/01/2021). We can thus use the following (monthly) observations on the ADS index: $X_s$, $s = 07/30/2020, 08/30/2020, 09/30/2020, 10/30/2020,$ and $11/30/2020$.



This sampling scheme is reminiscent of the unrestricted mixed data sampling proposed by Foroni et al. (2015), who showed that unrestricted MIDAS can outperform MIDAS when the difference in sampling frequencies between the regressand and the regressors is small. In our case, this mixed sampling approach facilitates application of various (nonlinear) ML and DL methods where it is not clear how to use distributed lag functions to avoid parameter proliferation due to the presence of a high-frequency predictor.

## 4.2 Out-of-Sample Performance across Methods

We compare the OoS performance of the forecasting methods described in Section 2. This subsection addresses the first question posed in the Introduction: *how does the GDP growth forecast performance of dense-modelling ML methods compare with that of sparse-modelling ML methods and other time series* *models in a sample with a large number of predictors with mixed predictive power or in a sample with a small number of strong predictors?*. We use the three sets of predictors defined in Section 4.1 to construct forecasts of U.S. GDP growth for four horizons, $h = 1, 2, 3, 4$ (quarters ahead). Note that we always include the first lag of U.S. GDP growth as an additional predictor.

*The large set of predictors Preds (i):* We compare the OoS performance of forecasts provided by the methods: LSTM, Ridge, Lasso, XGBoost, GBM, Random Forest, and Deep NN. First, we examine the RMSE values calculated from 142 sub-samples. The line plots in Figure 3 suggest that (1) the RMSE varies significantly across sub-samples, and it takes larger values for the sub-samples covering the Covid-19 period (around mid-2020); (2) the values of the RMSE for one-period ahead forecasts seem more stable than those for longer-horizon forecasts, thus the RMSE values have higher variance as the forecast horizon increases; and (3) Random Forest, XGBoost, GBM, and Lasso outperform the other methods. Moreover, to get an idea of the distributional properties of the RMSE values, we also present box plots (which consist of the median marked by the line within a box, the first and third quartiles which are the edges of the box, and the minimum and maximum individual RMSE values depicted by the two whiskers below and above the box respectively) in Figure 4. These box plots reveal that ($1'$) for one-period ahead forecast, the variance (measured by the *interquartile range* between the third and first quartiles) of the RMSE values is smallest for Lasso and XGBoost, and the median is lowest for Random Forest and Lasso; and ($2'$) for long-horizon forecast, the RMSE values of the Lasso forecasts have a greater variance than other methods. In this case, the RMSEs of both the GBM and Random Forest forecasts have lower variance, although they can have slightly higher medians than Lasso.

Next, we turn to the MAE values. The line plots in Figure 5 suggest that the MAE values behave quite similarly to the RMSE values across forecasting methods. There is a clear pattern that Random Forest, XGBoost, GBM, and Lasso outperform both Deep NN and Ridge. All of these methods also outperform LSTM. The box plots given in Figure 6 show the superior performance of Lasso, XGBoost, and Random Forest (which all have pretty similar patterns of MAE values) for one-period ahead forecast. However, Random Forest can outperform the other methods for long-horizon forecast.

Now, we examine the OoS $R^2$ values. The line plots shown in Figure 7 suggest that the OoS $R^2$ values of one-period ahead forecasts are generally stable across sub-samples while the OoS $R^2$ values of long-horizon forecasts show much more volatility. Random Forest, Lasso, and XGBoost still perform much better than the other algorithms. Moreover, the box plots given in Figure 8 indicate that the OoS $R^2$ values of one-period ahead forecasts by Lasso, XGBoost, GBM, and Random Forest have higher median and lower variance (*or* interquartile range). And most of the Random Forest OoS $R^2$ values are still higher than many OoS $R^2$ values provided by all the other methods. For long-horizon forecast, Random Forest can also provide better OoS $R^2$ values in terms of median and variance.

The key take-away from this experiment is that Random Forest appears to be the best forecasting method.

*The small set of strong predictors Preds (ii):* We compare the OoS forecast performance of the methods: LSTM, Ridge, Lasso (which is basically the OLS in this case as all the covariates are strong predictors, thus all get selected), sPCA, XGBoost, GBM, Random Forest, Deep NN, and Facebook Prophet. We start by inspecting the plots of the RMSE values. Figure 9 shows that the RMSE values of one-period ahead forecasts appear much more stable across sub-samples than those of long-horizon forecasts (except during the Covid-19 period), and Random Forest still performs quite well across forecast horizons while LSTM can yield lower RMSE values in several subperiods with the end dates ranging from 03/01/2009 to 03/01/2019. Moreover, the box plots in Figure 10 suggest that the RMSE values of one-period ahead forecasts produced by Ridge, Lasso, and Random Forest and those of long-horizon forecasts produced by sPCA and Random Forest can have lower medians than the other methods. The RMSE values of one-period ahead forecasts provided by XGBoost and GBM seem to have smaller variances while only the RMSE values of long-horizon forecasts by GBM have the lowest variance at the cost of higher medians.

Next, we turn to the MAE. Figure 11 shows that the MAE values appear quite unstable for all the forecast horizons while Deep NN and sPCA perform quite well for long-horizon forecast. The box plots in Figure 12 suggest that, for one-period forecast, Ridge and Lasso are the best methods immediately followed by sPCA and Random Forest. For long-horizon forecast, there is no clear cut winner.

Now, we examine the OoS $R^2$ values. The line plots given in Figure 13 suggest that the OoS $R^2$ values provided by Lasso, Ridge, sPCA, and Random Forest vary much less than the other methods. Therefore, the former methods can provide much more stable OoS forecast performance across different sub-samples. All the other methods, such as LSTM and GBM, can provide good forecasts only in specific sub-samples.

The box plots in Figure 14 suggest that, for one-period ahead forecast, Ridge, Lasso, sPCA, and Random Forest are the best performers with the Lasso OoS $R^2$ values achieving the lowest variance and the Ridge OoS $R^2$ values achieving the highest median. And the sPCA OoS $R^2$ values have lower median than the Random Forest OoS $R^2$ values. For two- or three-period ahead forecast, Random Forest can provide better OoS $R^2$ values than any other method as the interquartile range of the Random Forest OoS $R^2$ values seems smallest while the median is slightly greater than those by all other methods. sPCA and Deep NN also perform quite well (their OoS $R^2$ values are a few percentage points lower than the Random Forest OoS $R^2$ values).

The key take-aways from this second experiment is that one can employ Ridge or Lasso for *one-period ahead* forecast while sPCA and Random Forest are better for *long-horizon* forecast. This provides an evidence that gain from using nonlinear machine learning methods is not confined to a big data context.

*The small set of strong predictors plus the ADS index Preds (iii):* We compare the OoS forecast performance of the following ten methods: LSTM, Ridge, Lasso, sPCA, XGBoost, GBM, Random Forest, Deep NN, MIDAS, and Facebook Prophet. As before, we start by examining the OoS RMSE values. The line plots in Figure 15 suggest that the RMSE values of one-period ahead forecasts vary across sub-samples much less than those of long-horizon forecasts, except in the sub-samples including the Covid-19 period. Moreover, Random Forest, Deep NN, Ridge, Lasso, and MIDAS outperform all the other methods across all the forecast horizons. LSTM can only produce lower RMSE values for two- to four-period ahead forecasts in several sub-samples with the end dates between 06/01/2010 and 06/01/2020. The box plots in Figure 16 show that, for one-period ahead forecast, there seem to be not much difference in the median RMSE values provided by Ridge, Lasso, sPCA, Random Forest, and MIDAS, although the MIDAS RMSE values have the smallest variance (which is slightly smaller than the variance of the Ridge RMSE values). These methods are also the best performers. For long-horizon forecast, there is no clear-cut advantage among sPCA, Random Forest, and MIDAS; the RMSE values provided by both Ridge and Lasso can have slightly higher median and variance than those provided by sPCA.

Next, we examine the MAE values. Figure 17 shows that the MAE values vary significantly across sub-samples for all forecast horizons. Random Forest, Deep NN, sPCA, Ridge, and MIDAS still outperform all the other methods. The box plots in Figure 18 reveal that the MAE values of one-period ahead forecasts by

Ridge and MIDAS clearly have lower variances than those by Lasso, sPCA, and Random Forest (which all outperform LSTM, XGBoost, GBM, Deep NN, and Prophet). For long-horizon forecast, it is not obvious which method is the winner, although there is some evidence that Random Forest and MIDAS may still perform well.

We now turn to the OoS $R^2$ values. The line plots shown in Figure 19 suggest that Ridge, Lasso, sPCA, Random Forest, and MIDAS perform quite stably across sub-samples. Meanwhile, LSTM, XGBoost, GBM, and Prophet clearly underperform all of these methods. The box plots given in Figure 20 further reveal that the OoS $R^2$ values of one-period ahead forecasts by sPCA and Lasso have slightly lower medians but much lower variances than those by Ridge and MIDAS. The Ridge OoS $R^2$ values have a higher median than the MIDAS OoS $R^2$ values while their variances seem quite similar. For long-horizon forecast, MIDAS seems to outperform all the other methods as its OoS $R^2$ values have a slightly higher median and lower variance.

The key take-away from this third experiment is that, when one adds a strong high-frequency predictor to a set of high-quality low-frequency predictors, sPCA and Lasso can outperform MIDAS only for one-period ahead forecast while MIDAS can outperforms the other methods for long-horizon forecast.

## 4.3 Out-of-Sample Forecast Performance across Three Sets of Predictors

This subsection addresses the second and third questions posed in the Introduction: (ii) *which methods produce better forecasts with a large set of predictors than with a small subset of strong predictors?* and (iii) *which methods provide superior forecast performance overall across all the three different sets of predictors?* We compare the OoS forecast performance results obtained by using the three sets of predictors Preds (i), Preds (ii), and Preds (iii) (as defined in Section 4.1) for each of the following ten forecasting methods: Ridge, Lasso, XGBoost, GBM, Random Forest, Deep NN, LSTM, sPCA, Prophet, and MIDAS. We first examine the box plots of the RMSE values in Figure 21. For *one-period ahead* forecast, we can draw the following general conclusions:

- Ridge performs better with a few predictors because, as described in Section 2.2, Ridge regression does not select variables (and it just shrinks all the regression coefficients to very small values as a way to avoid the non-invertibility problem). Hence, having too many predictors with some very weak predictors may potentially increase the likelihood of overfitting.

- Lasso performs well for every set of predictors as it can select predictors that best predict the response variable automatically.

- XGBoost and GBM perform better with many predictors as these methods are ensembles of multiple regression trees where the tree grown in each iteration improves the error of the tree fitted in the previous iteration. Therefore, having many predictors facilitates growing many regression trees which may not have great predictive power individually, but a combination of them via boosting the prediction errors can potentially lead to good forecasts.

- For Random Forest and Deep NN, there is not much difference in their OoS forecast performance with a large set of predictors or with a small subset of strong predictors. Random Forest combines many independent regression trees while Deep NN optimally weights signals received by neurons at the input layer and hidden layers to predict the outcome variable (thus good predictors receive higher weights). On the other hand, LSTM usually performs better with a few good predictors because a LSTM model with mechanisms to remember or forget signals at previous points in time has much more parameters than a typical NN model, which makes LSTM more prone to the curse of dimensionality.

- The median of the RMSE values of forecasts by sPCA or Prophet can be improved by adding the ADS index to Preds (ii).

- Interestingly, MIDAS (using the ADS index as the sole predictor) can perform almost as well as Ridge or Lasso with Preds (iii).

For *long-horizon* forecast, MIDAS has a lower median RMSE than all the other methods while Random Forest using the large set Preds (i) has a slightly higher median RMSE, but lower variance, than MIDAS. Therefore, a simple linear method (like MIDAS) can provide better long-term forecasts than other sophisticated nonlinear ML methods if one has a good high-frequency predictor (such as the ADS index in this case). In reality, there may be more than one good high-frequency predictor that could accurately predict the GDP growth as well. We do not try to come up with the best possible forecasting MIDAS model here. However, we believe that additional high-frequency predictors would only strengthen the evidence for the superior OoS forecast performance of a parsimonious model like MIDAS in this forecast exercise.

Next, we turn to inspect the MAE values plotted in Figure 22. For *one-period ahead* forecast, Ridge, Lasso (using Preds (iii)), and MIDAS yield quite similar median MAE values. For long-horizon forecast, Random Forest (using Preds (i)) can outperform MIDAS in terms of MAE variance at the cost of higher median.

Finally, we compare the OoS $R^2$ values plotted in Figure 23. For one-period ahead forecast, the bottom-line results are:

- The OoS $R^2$ values provided by Lasso with Preds (iii) have smaller variance than Ridge with Preds (ii) at the cost of a little lower median.
- As with the RMSE values, the OoS $R^2$ values of XGBoost, GBM, and Random Forest are higher with a large set of predictors. sPCA also performs much better with Preds (iii).
- The OoS $R^2$ values provided by MIDAS have the same median as those provided by Lasso and sPCA, but they have a slightly higher variance than those provided by Lasso.

The key take-away from this one-period ahead forecast exercise is that many methods (such as Ridge, Lasso, sPCA, XGBoost, GBM, and Random Forest) can yield quite similar OoS $R^2$ to MIDAS. If we want a good one-period ahead forecast with low variance, then sPCA can be a good choice in case there is a good high-frequency predictor, otherwise the RF can be used. On the other hand, the box plots of the OoS $R^2$ values of long-horizon forecasts suggest that sPCA, Random Forest, and MIDAS have roughly the same median OoS $R^2$ values, but MIDAS seems to be a good choice in this case (especially, when the forecasted variable is quite volatile) because its OoS $R^2$ values have smaller variance than those of the other two methods.

## 5 Conclusion

Forecasting macroeconomic variables is an important task in economics as forecasts of these variables are often used by economic agents to form their expectation and by policy makers to understand the current and future state of the economy. This paper evaluates and compares the OoS performance of many popular forecasting methods employed to forecast U.S. GDP growth with a large/small set of predictors. Our main take-aways drawn from this forecast experiment are: (1) when one uses a large set of predictors with mixed predictive power, there is no clear-cut advantage of using a density-based ML method over a sparsity-based ML method for one-period ahead forecast, while there is a clear advantage of using a density-based ML method (such as Random Forest) for long-term forecast; (2) when one uses a small subset of strong predictors, the OLS method can outperform density-based ML methods for one-period ahead forecast, while there is no significant difference in the OoS forecast performance between a factor-augmented regression (i.e., sPCA) and a density-based ML method (i.e., Random Forest) for long-horizon forecast; (3) density-based ML methods could perform better with a large set of predictors than with a small set of strong predictors; (4) when one has a good high-frequency predictor, simple forecasting methods (e.g., sPCA and MIDAS) can provide better forecasts in the short and long horizons respectively than many sophisticated ML and DL methods, highlighting the important role of predictors in economic forecasting; and finally (5) we also demonstrate that ML methods can outperform DL methods.

## References

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

Athey, S. and G. W. Imbens (2019). Machine learning methods economists should know about. *Annual Review of Economics 11*, 685–725.

Baffigi, A., R. Golinelli, and G. Parigi (2004). Bridge models to forecast the euro area GDP. *International Journal of Forecasting 20*, 447–460.

Bair, E., T. Hastie, D. Paul, and R. Tibshirani (2006). Prediction by supervised principal components. *Journal of the American Statistical Association 101*(473), 119–137.

Barkan, O., J. Benchimol, I. Caspi, A. Hammer, and N. Koenigstein (2021). Forecasting CPI inflation components with hierarchical recurrent neural network. mimeo.

Bengio, Y., P. Frasconi, and P. Simard (1993). The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, San Francisco, pp. 1183–1195. IEEE Press.

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

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Carriero, A., A. B. Galvão, and G. Kapetanios (2019). A comprehensive evaluation of macroeconomic forecasting methods. *International Journal of Forecasting 35*(4), 1226–1239.

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

Dahl, C. M. and E. N. Sørensen (2021, February). Time series (re)sampling using generative adversarial networks. mimeo.

Foroni, C., M. Marcellino, and C. Schumacher (2015). U-MIDAS: MIDAS regressions with unrestricted lag polynomials. *Journal of the Royal Statistical Society. Series A 178*(1), 57–82.

Ghysels, E., P. Santa-Clara, and R. Valkanov (2004). The MIDAS touch: mixed data sampling regression models. mimeo, https://www.cirano.qc.ca/files/publications/2004s-20.pdf.

Ghysels, E., A. Sinko, and R. Valkanov (2007). MIDAS regressions: further results and new directions. *Econometric Review 26*(1), 53–90.

Giacomini, R. and B. Rossi (2010). Forecast comparisons in unstable environments. *Journal of Applied Econometrics 25*(4), 595–620.

Giannone, D., M. Lenza, and G. E. Primiceri (2021). Economic predictions with big data: the illusion of sparsity. *Econometrica* (forthcoming).

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Götz, T. B. and T. A. Knetsch (2019). Google data in bridge equation models for German GDP. *International Journal of Forecasting 35*(1), 45–66.

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

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (second ed.). Springer.

Ho, T. K. (1995). Random decision forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition 1*(1), 278–282. Montreal, QC, Canada.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural Computation 9*(8), 1735–1780.

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

Huber, F., G. Koop, and L. Onorante (2021). Inducing sparsity and shrinkage in time-varying parameter models. *Journal of Business & Economic Statistics 39*(3), 669–683.

Kingma, D. P. and J. Ba (2017). Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980.

Lahiri, S. N. (2003). *Resampling Methods for Dependent Data*. Springer.

McAlinn, K. (2021). Mixed-frequency bayesian predictive synthesis for economic nowcasting. *Journal of the Royal Statistical Society: Series C* (forthcoming).

McCracken, M. W. and S. Ng (2020, March). FRED-QD: a quarterly database for macroeconomic research. working paper, `https://research.stlouisfed.org/wp/more/2020-005`.

Medeiros, M. C., G. F. R. Vasconcelos, A. Veiga, and E. Zilberman (2021). Forecasting inflation in a data-rich environment: The benefits of machine learning methods. *Journal of Business & Economic Statistics 39*(1), 98–119.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nakamura, E. (2005). Inflation forecasting using a neural network. *Economics Letters 86*(3), 373–378.

Paranhos, L. (2021). Predicting inflation with neural networks. mimeo.

Schorfheide, F. and D. Song (2015). Real-time forecasting with a mixed-frequency VAR. *Journal of Business & Economic Statistics 33*(3), 366–380.

Stock, J. H. and M. W. Watson (2002a). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association 97*(460), 1167–1179.

Stock, J. H. and M. W. Watson (2002b). Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics 20*(2), 147–162.

Taylor, S. J. and B. Letham (2018). Forecasting at scale. *American Statistician 72*(1), 37–45.

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

Woody, S., C. M. Carvalho, and J. S. Murray (2021). Model interpretation through lower-dimensional posterior summarization. *Journal of Computational and Graphical Statistics 30*(1), 144–161.

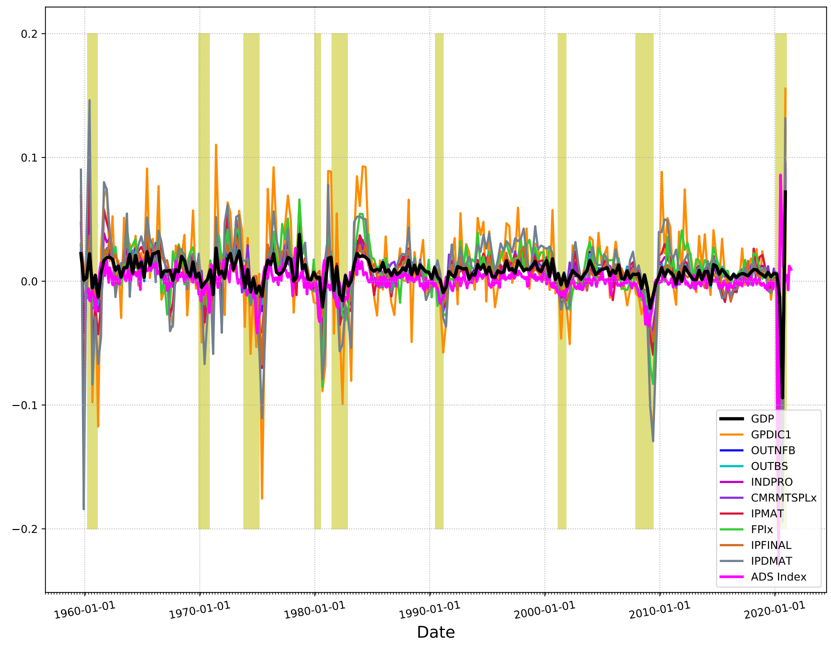
Yousuf, K. and Y. Feng (2021). Targeting predictors via partial distance correlation with applications to financial forecasting. *Journal of Business & Economic Statistics* (forthcoming).

## A.I Tables and Figures

Table 1: Variables pre-selected from FRED-QD by Yousuf and Feng's (2021) variable screening procedure based on the partial distance correlation. (All these variables were transformed using the function $\Delta \log(x_t)$)

| FRED Mnemonic | Description |
|---|---|
| GPDIC1 | Real Gross Private Domestic Investment, 3 decimal (Billions of Chained 2012 Dollars) |
| OUTNFB | Nonfarm Business Sector: Real Output (Index 2012=100) |
| OUTBS | Business Sector: Real Output (Index 2012=100) |
| INDPRO | Industrial Production Index (Index 2012=100) |
| CMRMTSPLx | Real Manufacturing and Trade Industries Sales (Millions of Chained 2012 Dollars) |
| IPMAT | Industrial Production: Materials (Index 2012=100) |
| FPIx | Real private fixed investment (Billions of Chained 2012 Dollars), deflated using PCE |
| IPFINAL | Industrial Production: Final Products (Market Group) (Index 2012=100) |
| IPDMAT | Industrial Production: Durable Materials (Index 2012=100) |

Figure 2: Plots of U.S. GDP growth rates versus the nine variables listed in Table 1 (above) and Aruoba et al.'s (2009) (ADS) business condition index

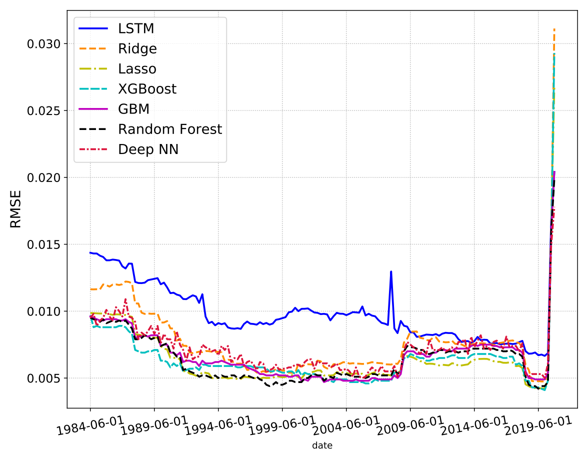


*The shaded bars are the NBER recession dates.

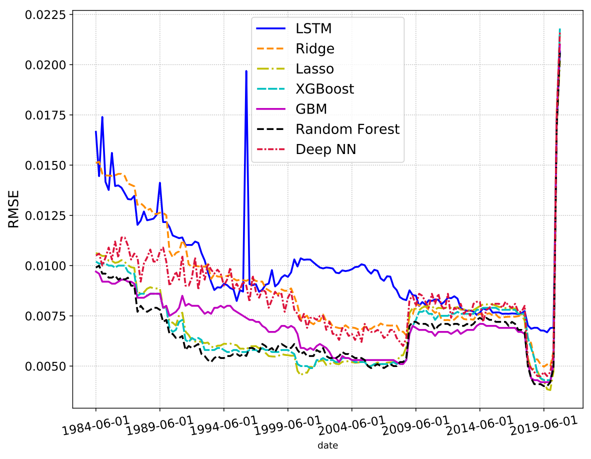
Table 2: Variables removed from FRED-QD due to many missing observations

| FRED Mnemonic | Description |
|---|---|
| MORTGAGE30US | 30-Year Conventional Mortgage Rate (Percent) |
| SPCS10RSA | S&P/Case-Shiller 10-City Composite Home Price Index (Index January 2000 = 100) |
| HOAMS | Manufacturing Sector: Hours of All Persons (Index 2012=100) |
| LNS13023557 | Unemployment Level - Reentrants to Labor Force (Thousands of Persons) |
| LNS13023705 | Unemployment Level - Job Leavers (Thousands of Persons) |
| USEPUINDXM | Economic Policy Uncertainty Index for United States |
| LNS13023621 | Unemployment Level - Job Losers (Thousands of Persons) |
| LNS13023569 | Unemployment Level - New Entrants (Thousands of Persons) |
| ACOGNOx | Real Value of Manufacturers' New Orders for Consumer Goods Industries (Millions of 2012 Dollars), deflated by Core PCE |
| TWEXAFEGSMTHx | Trade Weighted U.S. Dollar Index: Advanced Foreign Currencies (Index Jan 2006=100) |
| IMFSLx | Real Institutional Money Funds (Billions of 2012 Dollars), deflated by Core PCE |
| USSTHPI | All-Transactions House Price Index for the United States (Index 1980 Q1=100) |
| INVCQRMTSPL | Real Manufacturing and Trade Inventories (Millions of 2012 Dollars) |
| OUTMS | Manufacturing Sector: Real Output (Index 2012=100) |
| TCU | Capacity Utilization: Total Industry (Percent of Capacity) |
| DRIWCIL | FRB Senior Loans Officer Opions. Net Percentage of Domestic Respondents Reporting Increased Willingness to Make Consumer Installment Loans |
| ANDENOx | Real Value of Manufacturers' New Orders for Capital Goods: Nondefense Capital Goods Industries (Millions of 2012 Dollars), deflated by Core PCE |
| ULCMFG | Manufacturing Sector: Unit Labor Cost (Index 2012=100) |
| EXUSEU | U.S. / Euro Foreign Exchange Rate (U.S. Dollars to One Euro) |
| COMPRMS | Manufacturing Sector: Real Compensation Per Hour (Index 2012=100) |
| SPCS20RSA | S&P/Case-Shiller 20-City Composite Home Price Index (Index January 2000 = 100) |
| OPHMFG | Manufacturing Sector: Real Output Per Hour of All Persons (Index 2012=100) |
| MORTG10YRx | 30-Year Conventional Mortgage Rate Relative to 10-Year Treasury Constant Maturity (Percent) |

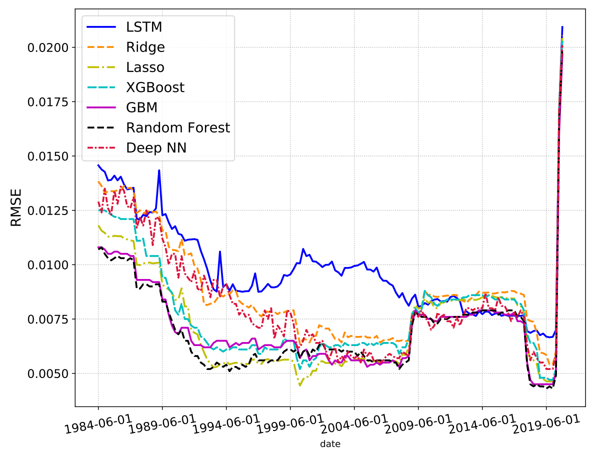
Figure 3: Line plots of OoS RMSEs of forecasts based on a large set of predictors (Preds (i))



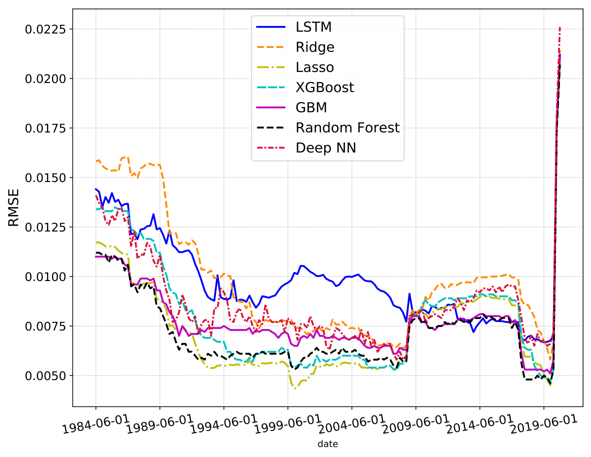
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

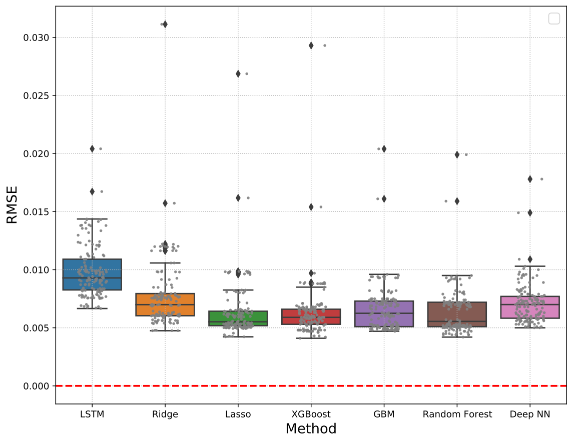


(d) four-period ahead forecasts

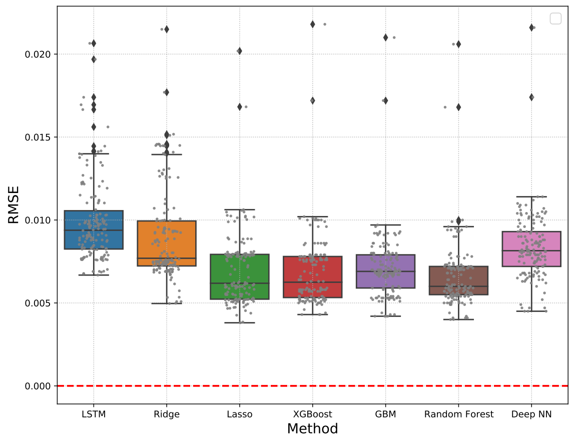
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

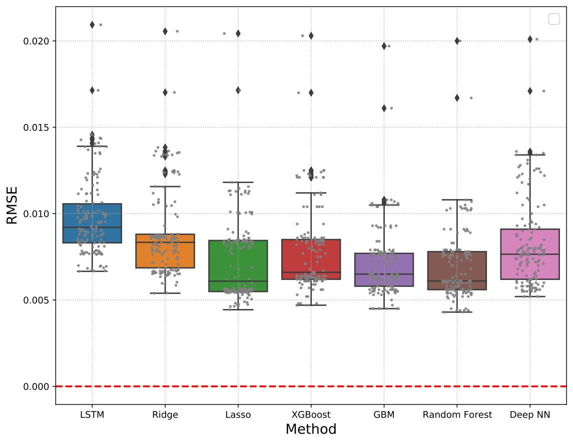
Figure 4: Box plots of OoS RMSEs of forecasts based on a large set of predictors (Preds (i))



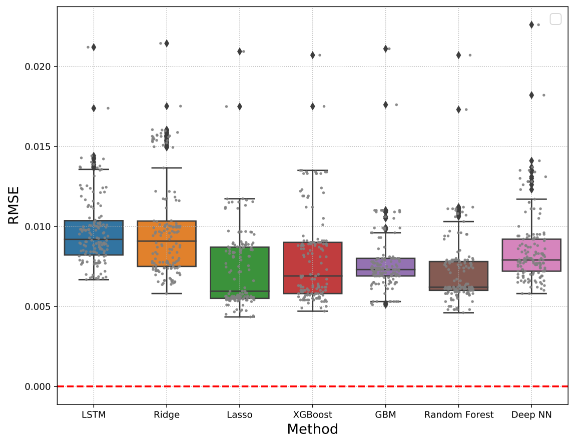
(a) one-period ahead forecasts



(b) two-period ahead forecasts



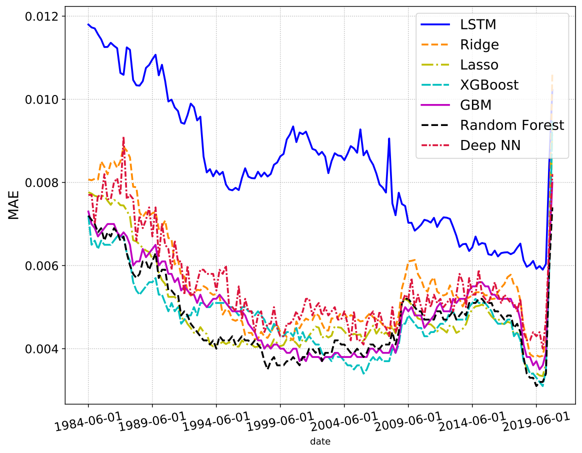
(c) three-period ahead forecasts



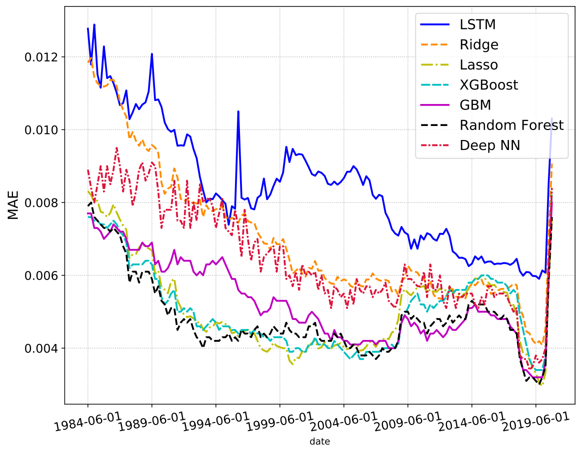
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

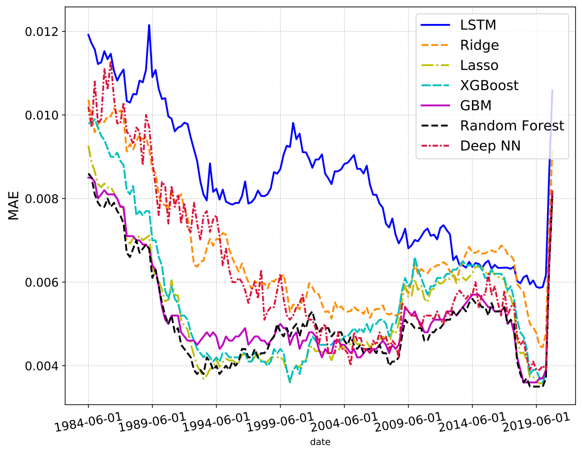
Figure 5: Line plots of OoS MAEs of forecasts based on a large set of predictors (Preds (i))



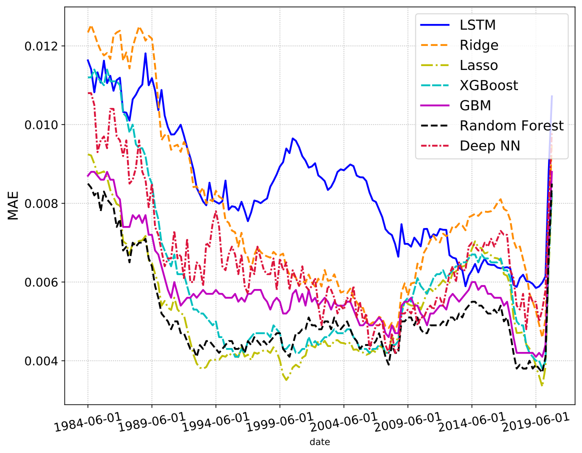
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

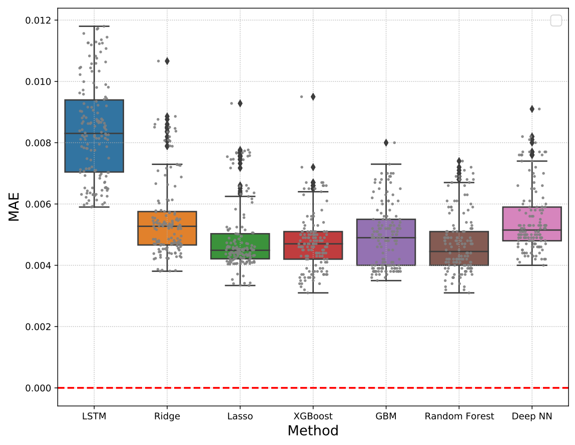


(d) four-period ahead forecasts

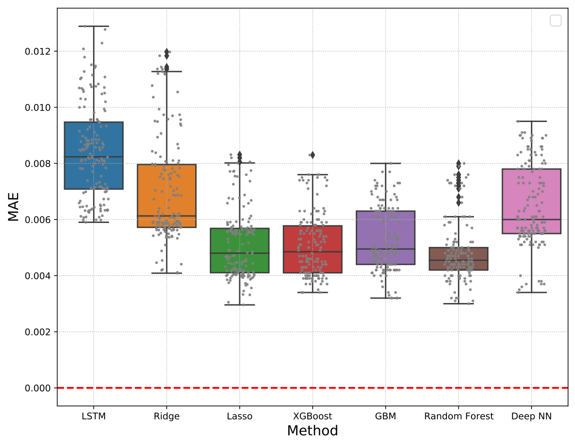
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

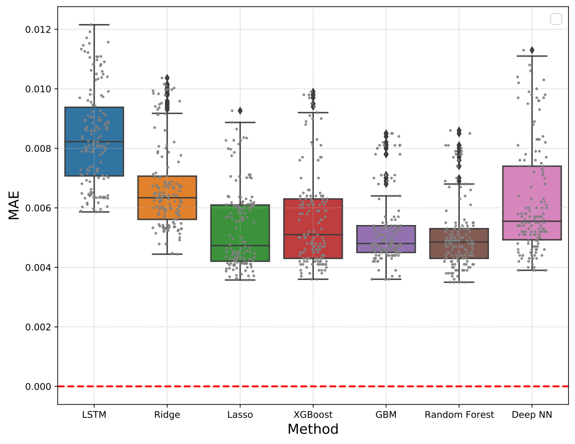
Figure 6: Box plots of OoS MAEs of forecasts based on a large set of predictors (Preds (i))



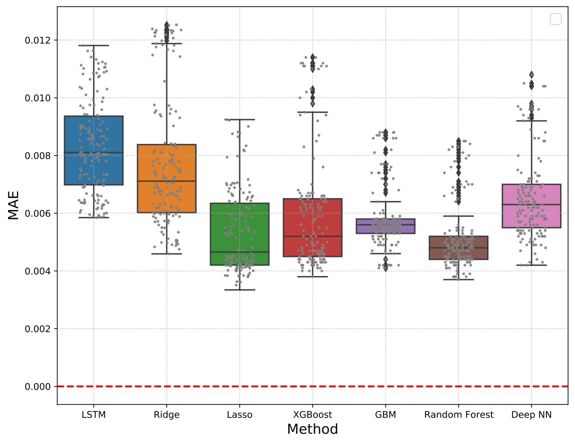
(a) one-period ahead forecasts



(b) two-period ahead forecasts



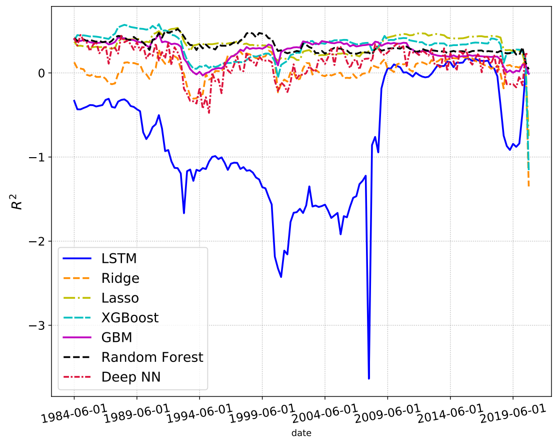
(c) three-period ahead forecasts



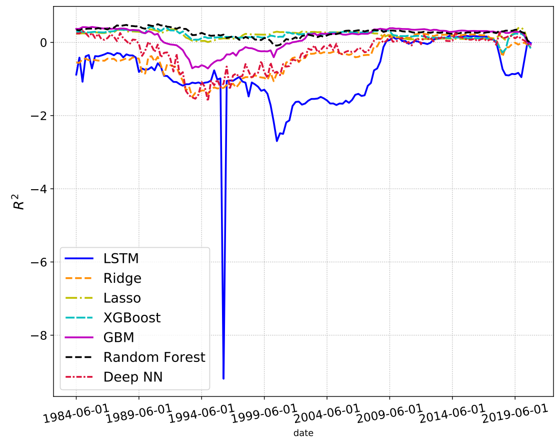
(d) four-period ahead forecasts

[*] The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

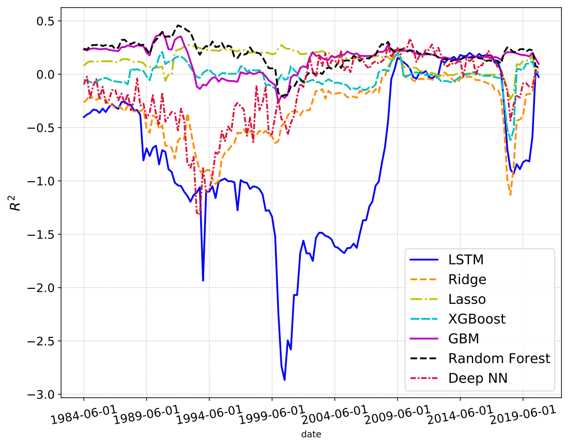
Figure 7: Line plots of OoS $R^2$ of forecasts based on a large set of predictors (Preds (i))



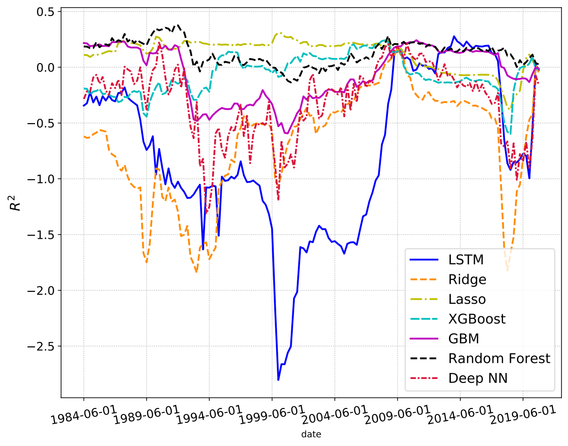
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

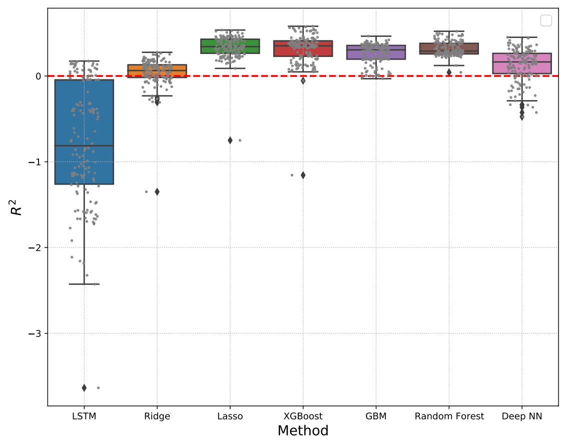


(d) four-period ahead forecasts

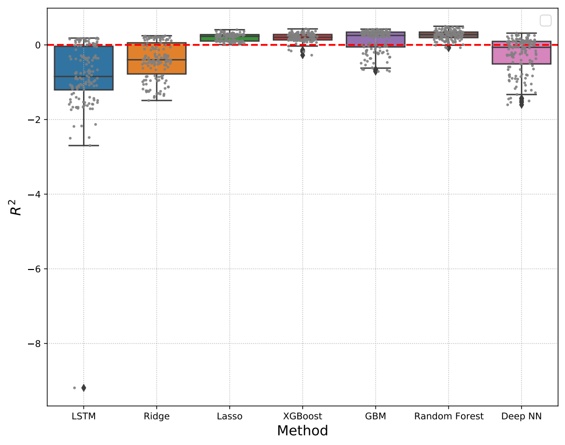
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

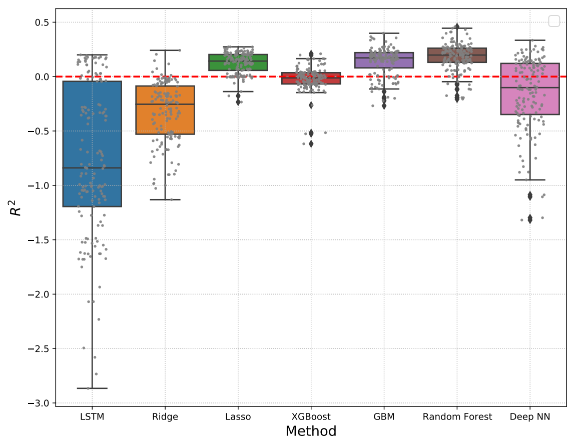
Figure 8: Box plots of OoS $R^2$ of forecasts based on a large set of predictors (Preds (i))



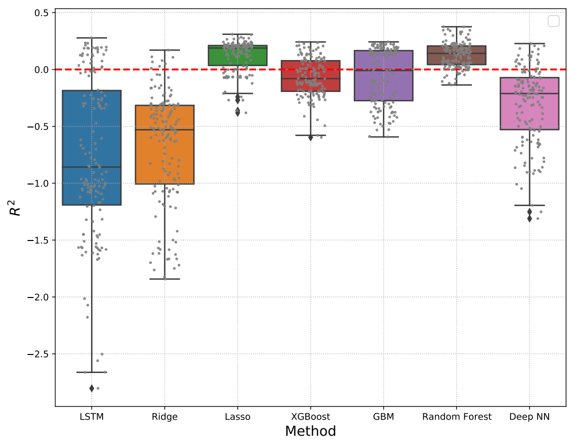
(a) one-period ahead forecasts



(b) two-period ahead forecasts



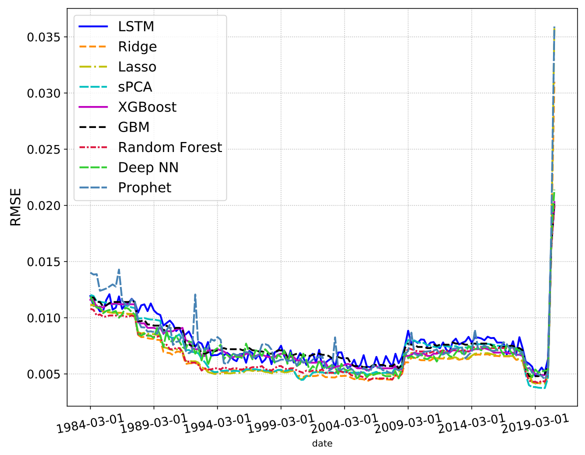
(c) three-period ahead forecasts



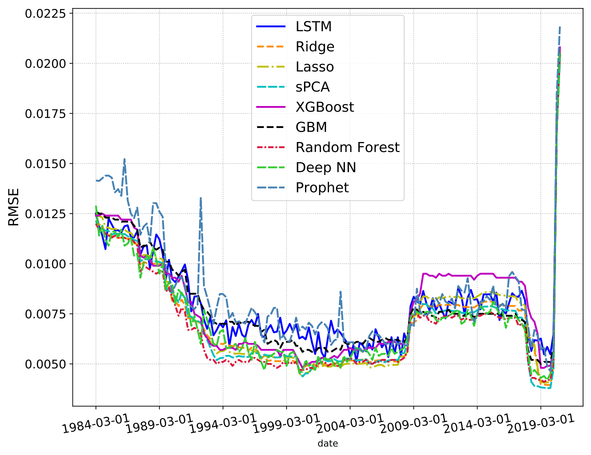
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

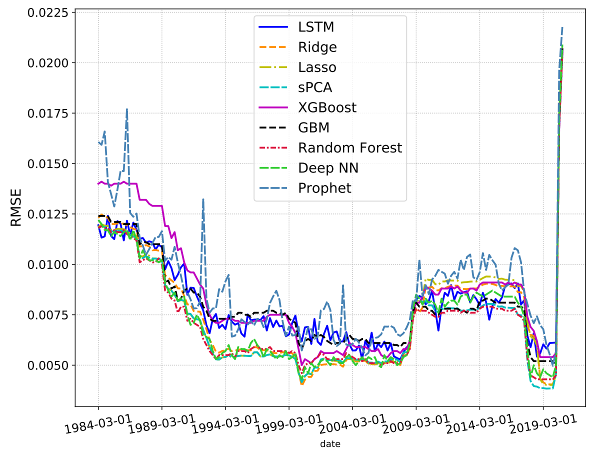
Figure 9: Line plots of OoS RMSEs of forecasts based on a small set of strong predictors (Preds (ii))



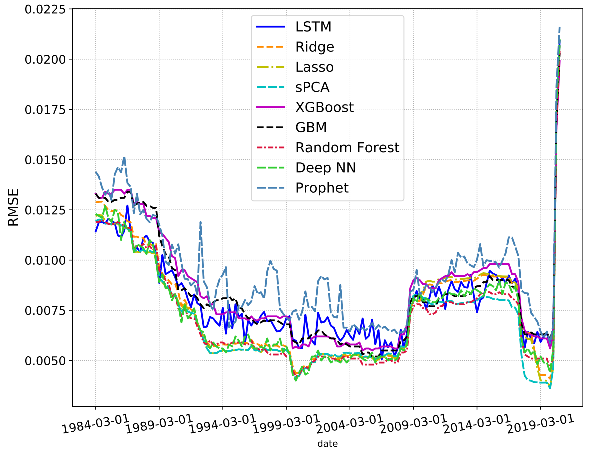
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

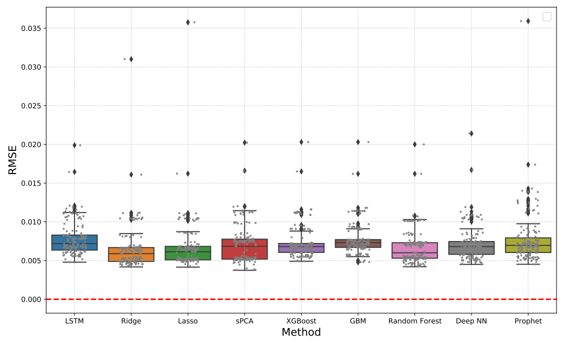


(d) four-period ahead forecasts

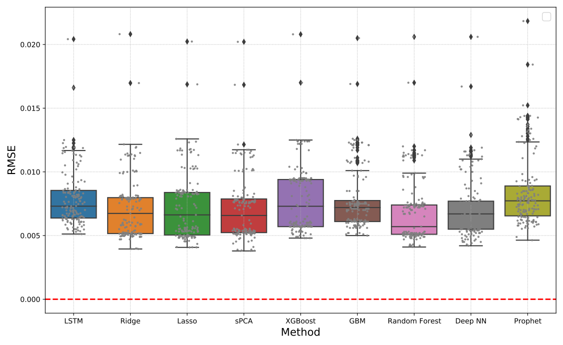
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

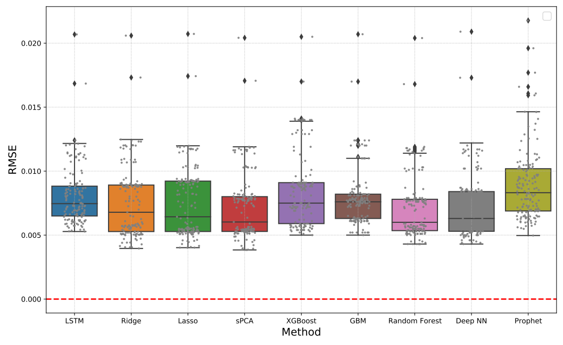
Figure 10: Box plots of OoS RMSEs of forecasts based on a small set of strong predictors (Preds (ii))



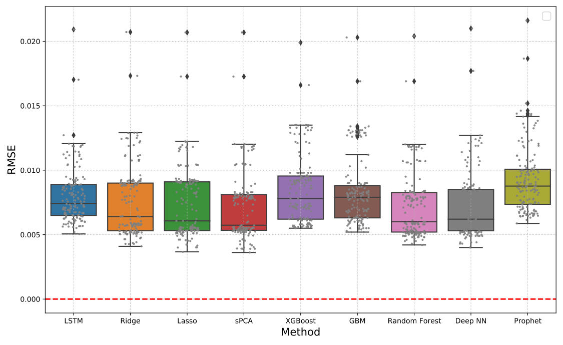
(a) one-period ahead forecasts



(b) two-period ahead forecasts



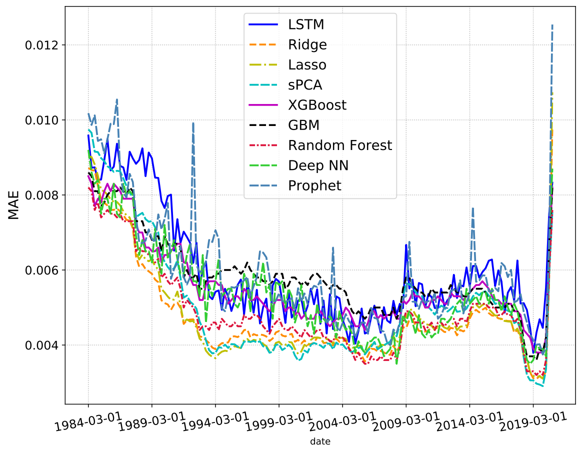
(c) three-period ahead forecasts



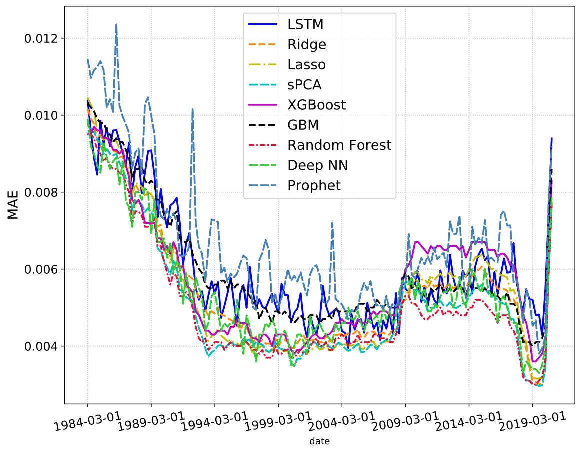
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

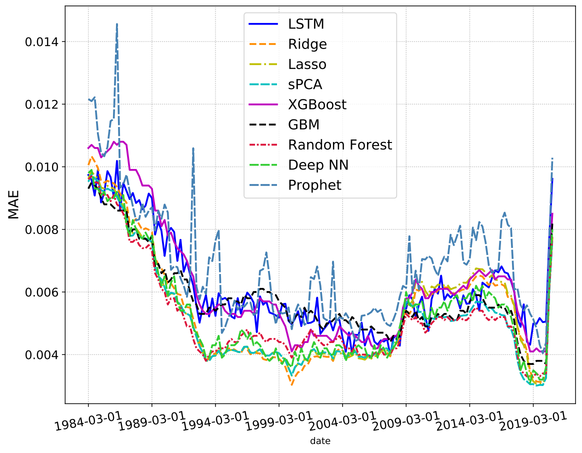
Figure 11: Line plots of OoS MAEs of forecasts based on a small set of strong predictors (Preds (ii))



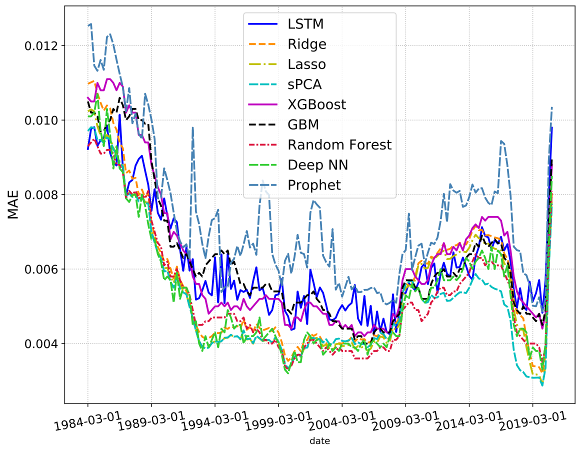
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

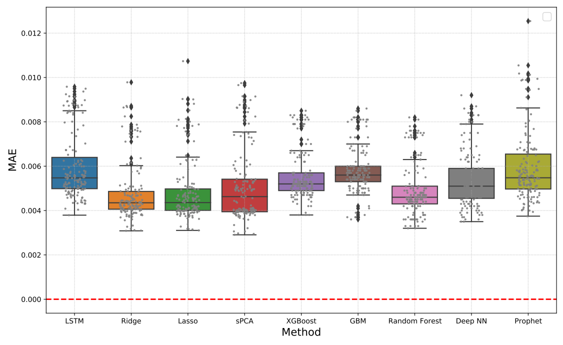


(d) four-period ahead forecasts

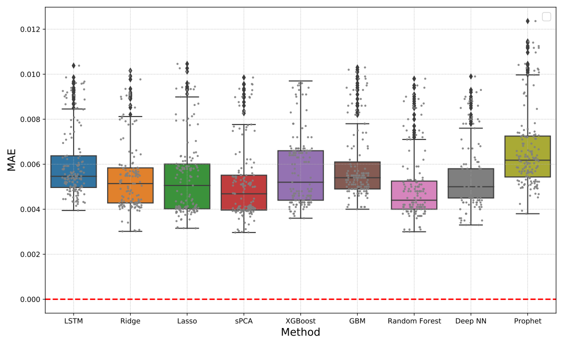
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

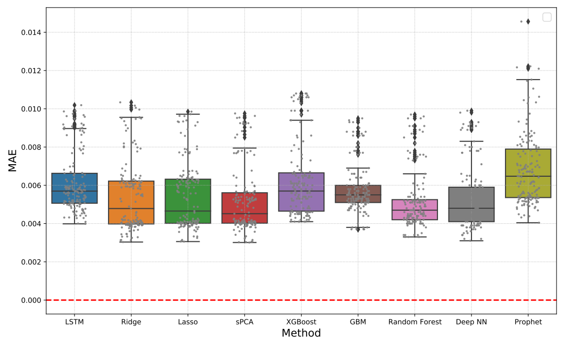
Figure 12: Box plots of OoS MAEs of forecasts based on a small set of strong predictors (Preds (ii))



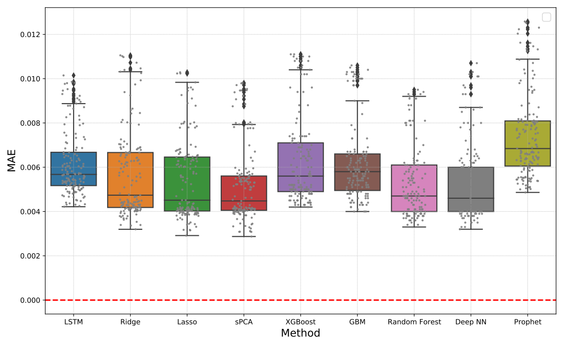
(a) one-period ahead forecasts



(b) two-period ahead forecasts



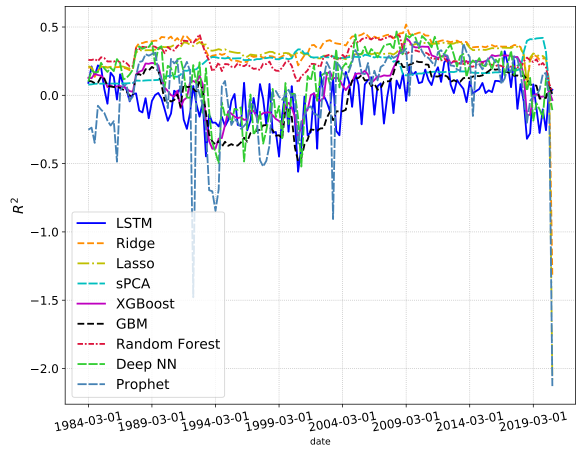
(c) three-period ahead forecasts



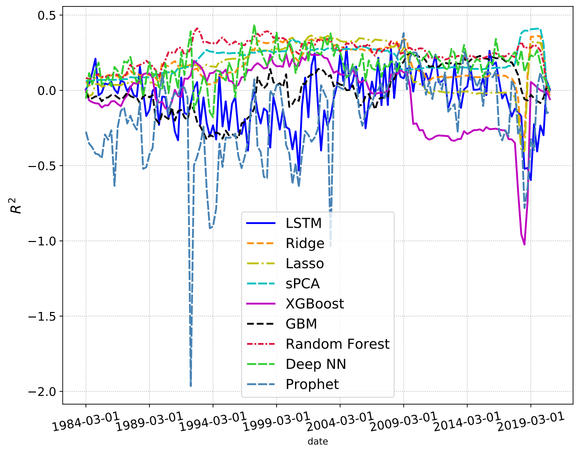
(d) four-period ahead forecasts

[*] The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

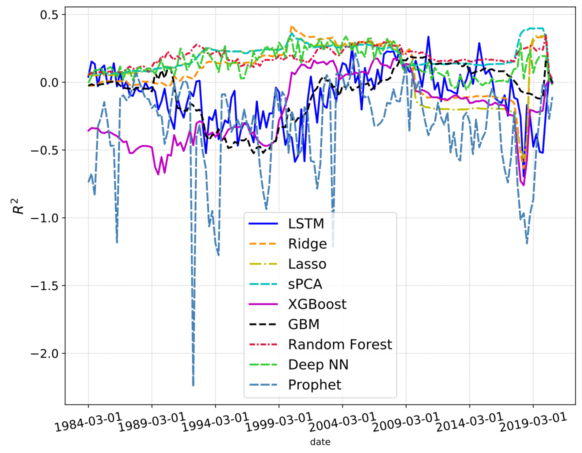
Figure 13: Line plots of OoS $R^2$ of forecasts based on a small set of strong predictors (Preds (ii))



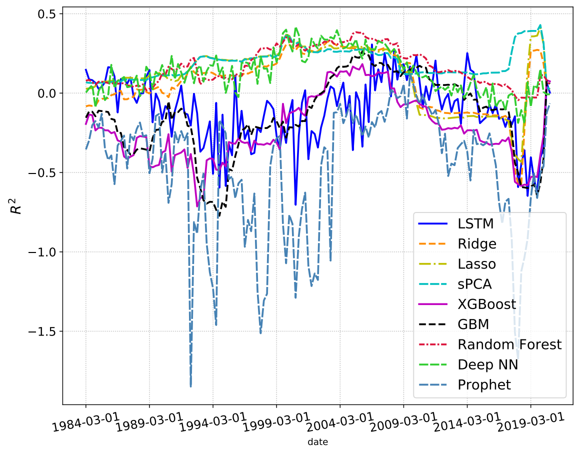
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

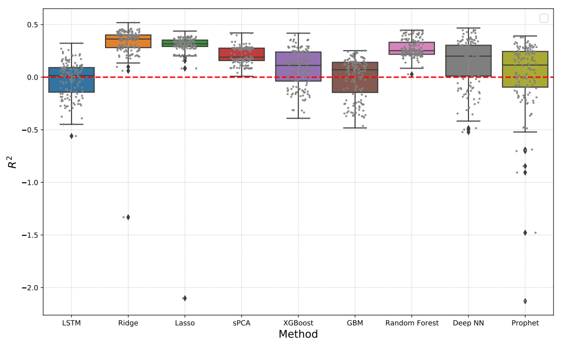


(d) four-period ahead forecasts

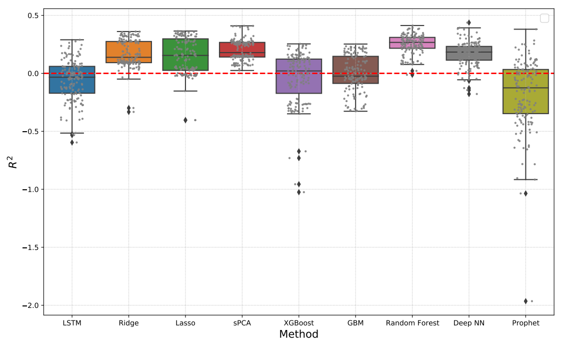
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

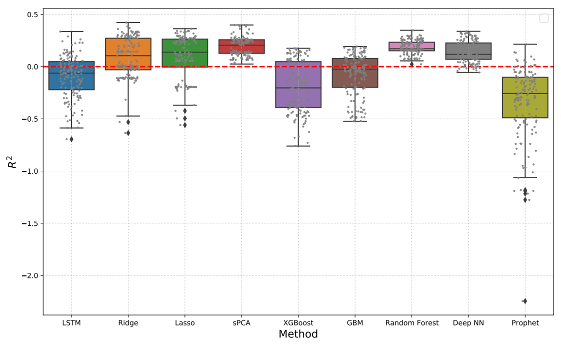
Figure 14: Box plots of OoS $R^2$ of forecasts based on a small set of strong predictors (Preds (ii))



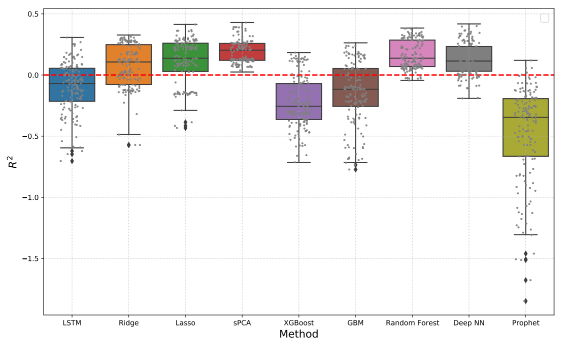
(a) one-period ahead forecasts



(b) two-period ahead forecasts



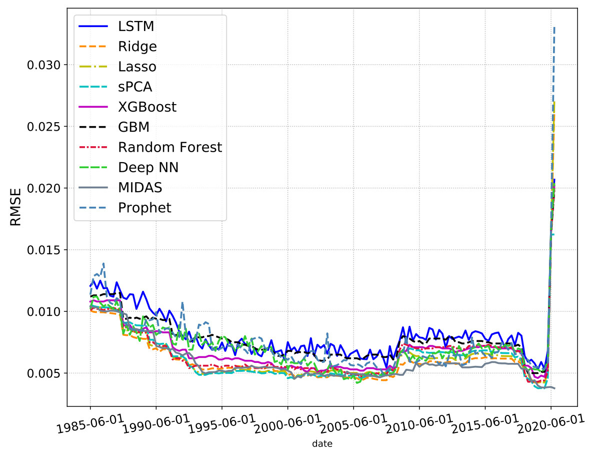
(c) three-period ahead forecasts



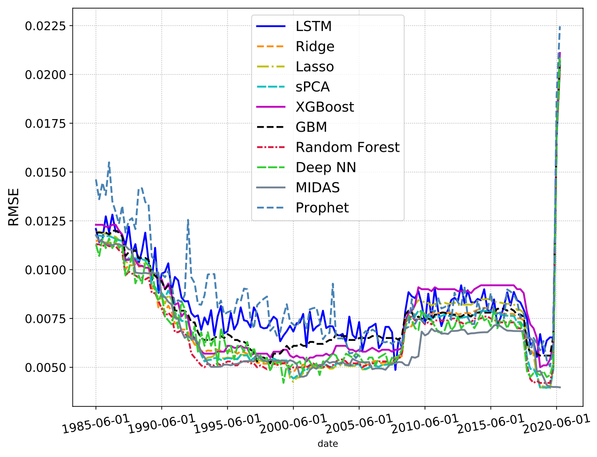
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

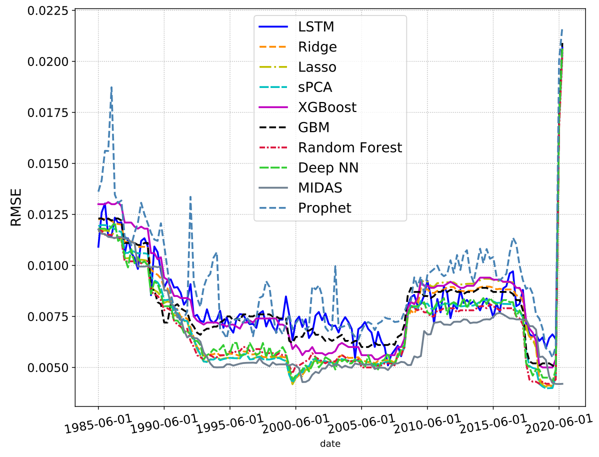
Figure 15: Line plots of OoS RMSEs of forecasts based on a small set of strong predictors plus the ADS index (Preds (iii))



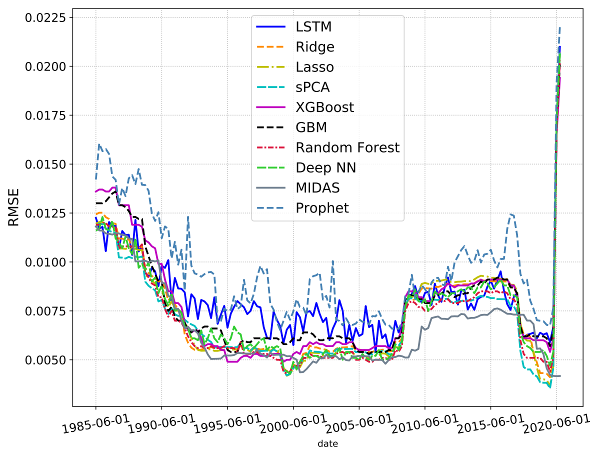
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

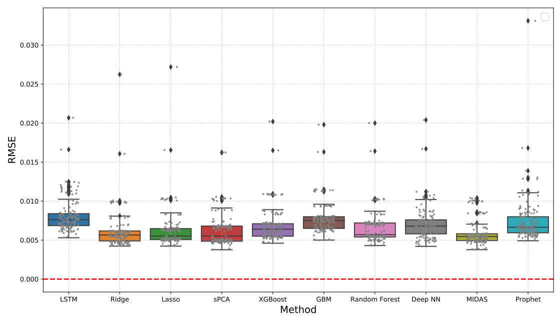


(d) four-period ahead forecasts

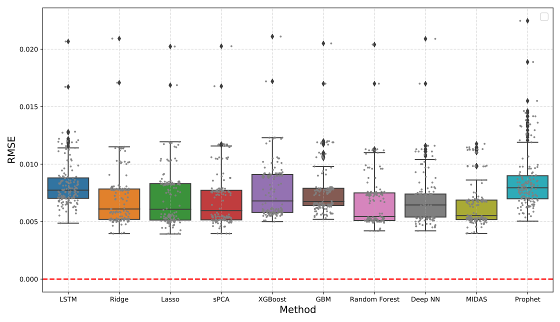
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

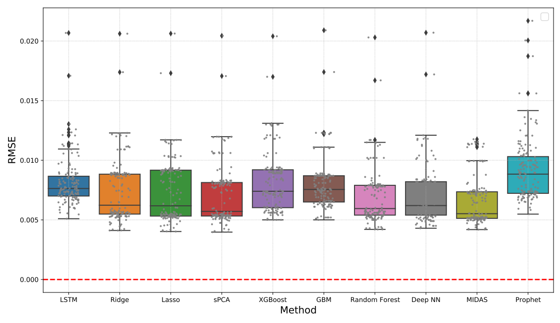
Figure 16: Box plots of OoS RMSEs of forecasts based on a small set of strong predictors plus the ADS index (Preds (iii))



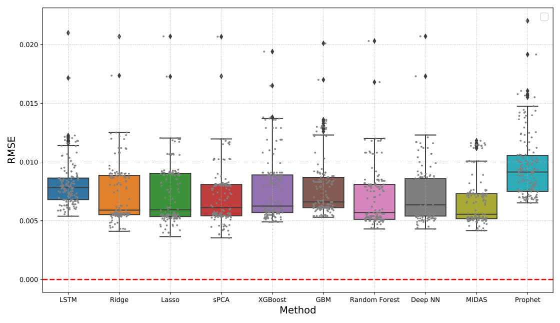
(a) one-period ahead forecasts



(b) two-period ahead forecasts



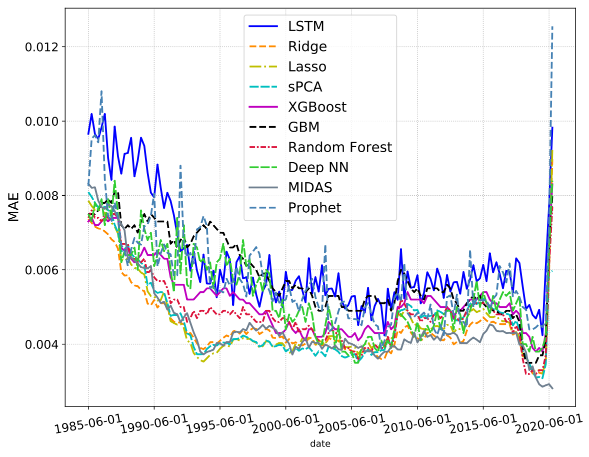
(c) three-period ahead forecasts



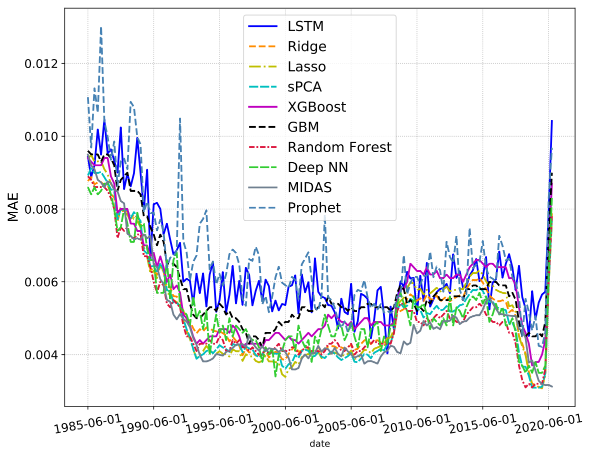
(d) four-period ahead forecasts

[*] The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

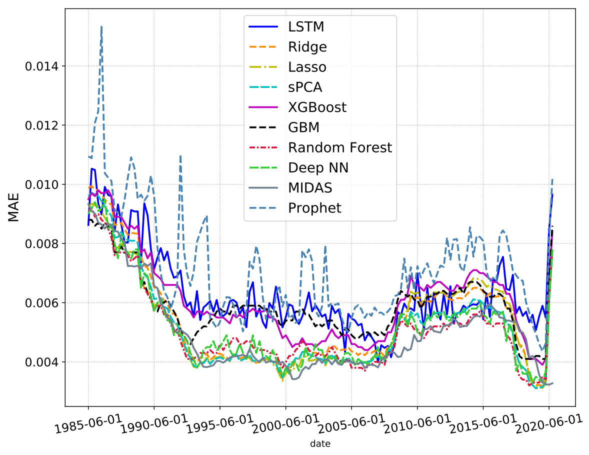
Figure 17: Line plots of OoS MAEs of forecasts based on a small set of strong predictors plus the ADS index (Preds (iii))



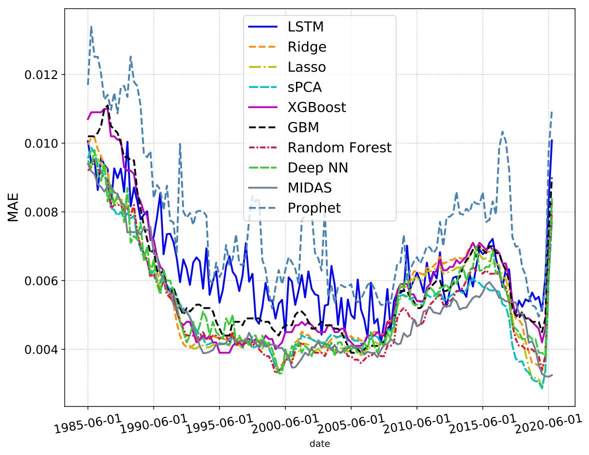
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

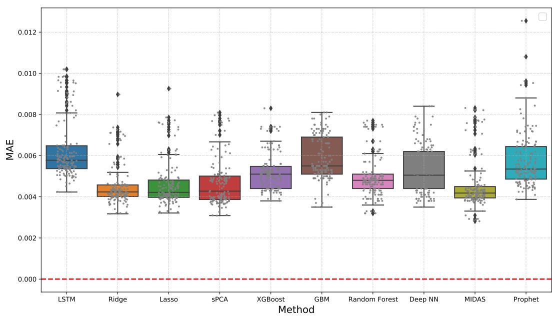


(d) four-period ahead forecasts

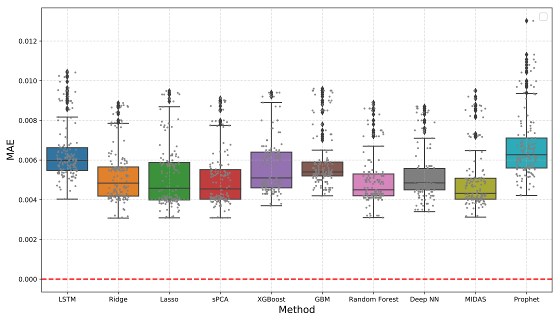
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

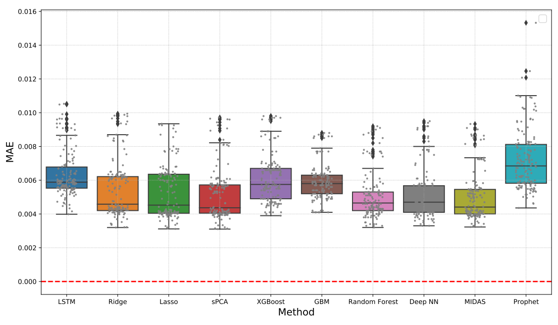
Figure 18: Box plots of OoS MAEs of forecasts based on a small set of strong predictors plus the ADS index (Preds (iii))



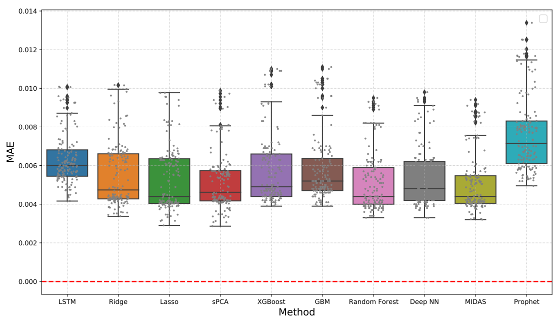
(a) one-period ahead forecasts



(b) two-period ahead forecasts



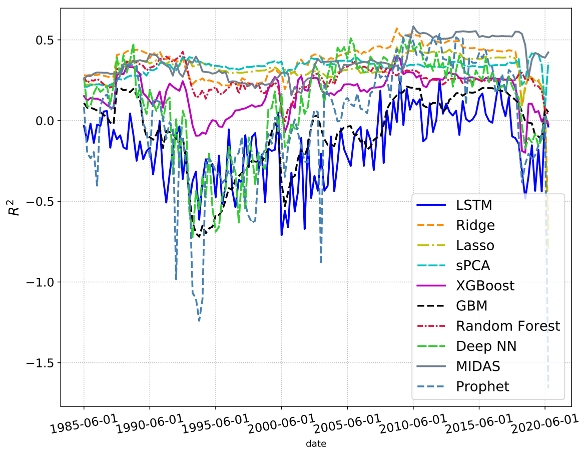
(c) three-period ahead forecasts



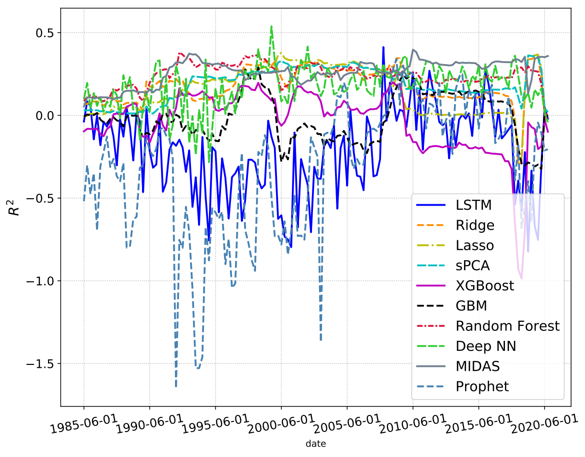
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

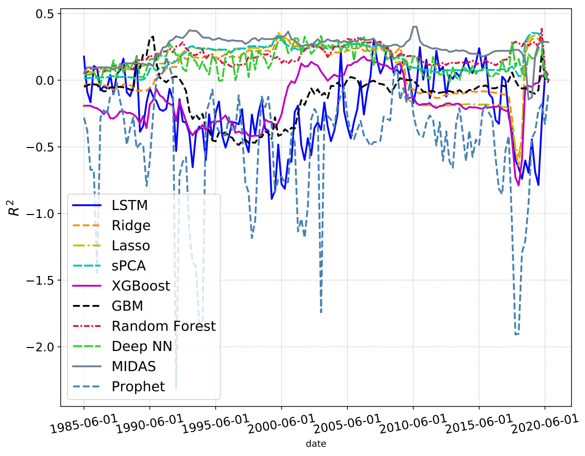
Figure 19: Line plots of OoS $R^2$ of forecasts based on a small set of strong predictors plus the ADS index (Preds (iii))



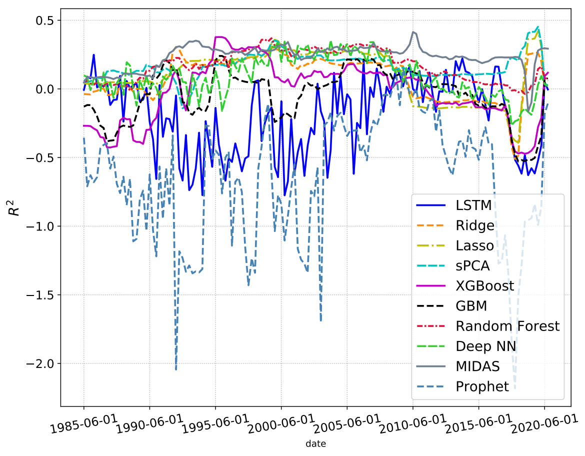
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts

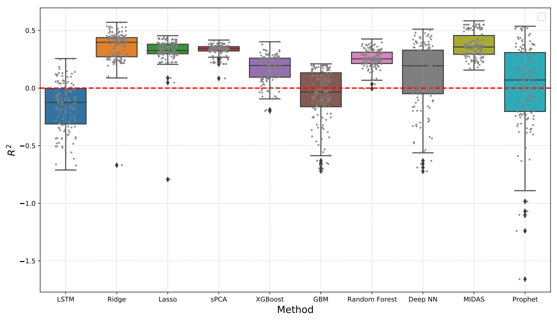


(d) four-period ahead forecasts

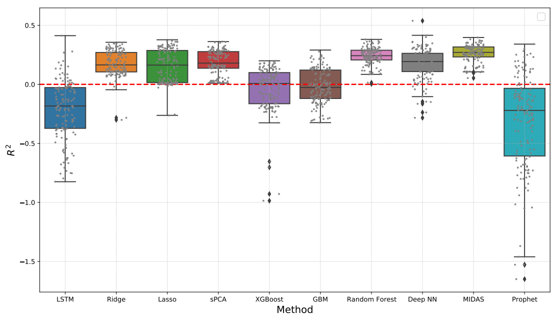
* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

* Dates appearing on the horizonal axis are the end dates of sub-samples.

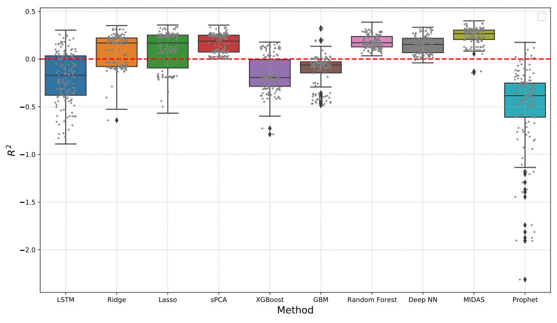
Figure 20: Box plots of OoS $R^2$ of forecasts based on a small set of strong predictors plus the ADS index (Preds (iii))



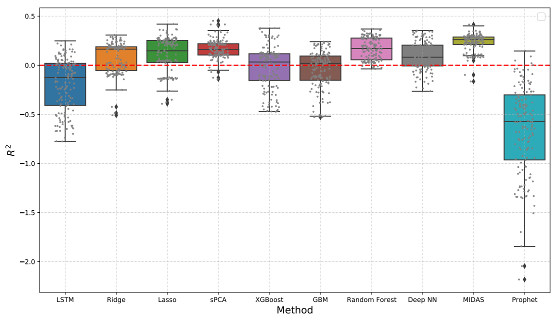
(a) one-period ahead forecasts



(b) two-period ahead forecasts



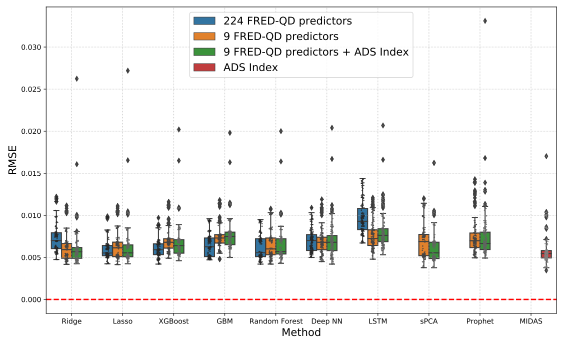
(c) three-period ahead forecasts



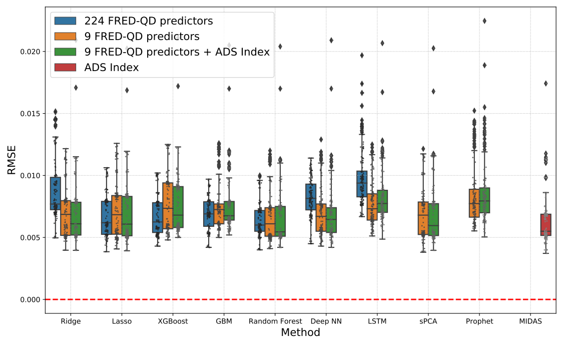
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

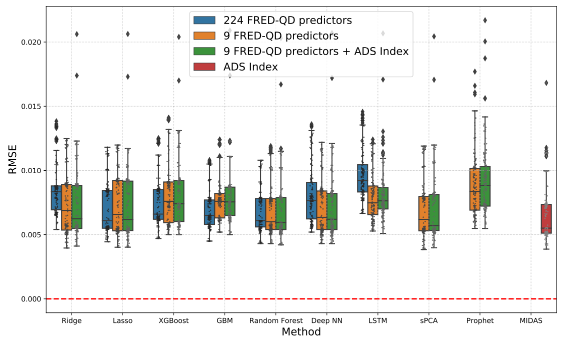
Figure 21: Box plots of OoS RMSEs of forecasts produced by various methods across three different sets of predictors (Preds (i), Preds (ii), and Preds (iii))



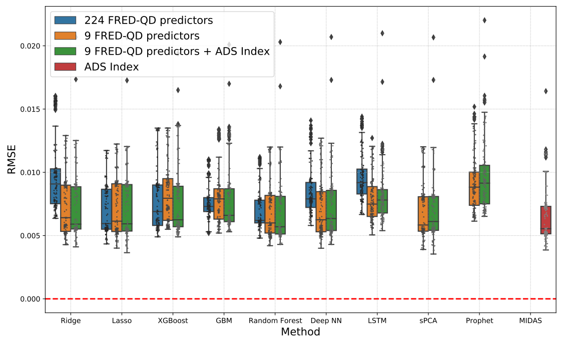
(a) one-period ahead forecasts



(b) two-period ahead forecasts



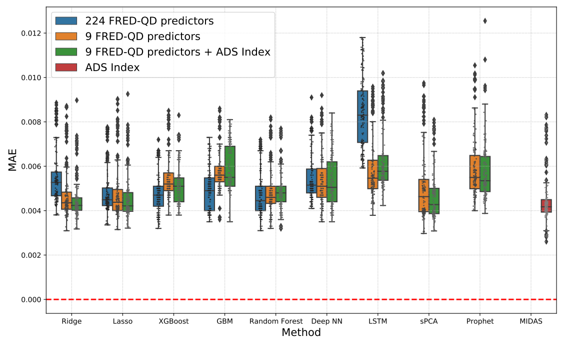
(c) three-period ahead forecasts



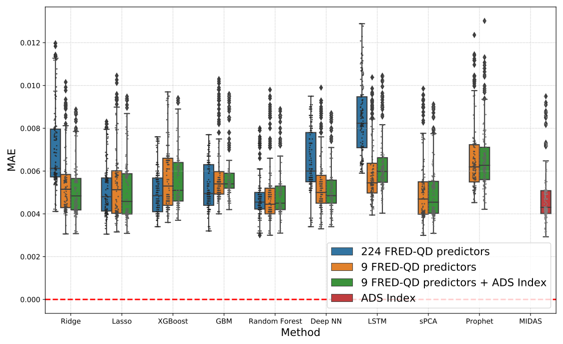
(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

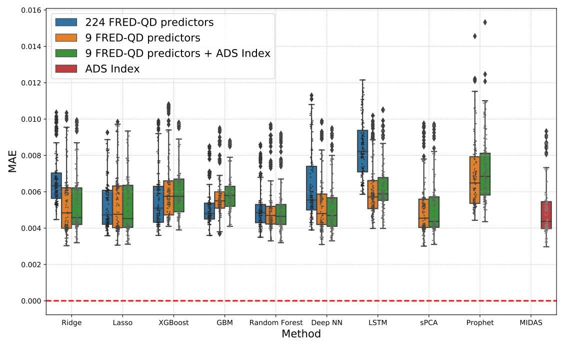
Figure 22: Box plots of OoS MAEs of forecasts produced by various methods across three different sets of predictors (Preds (i), Preds (ii), and Preds (iii))



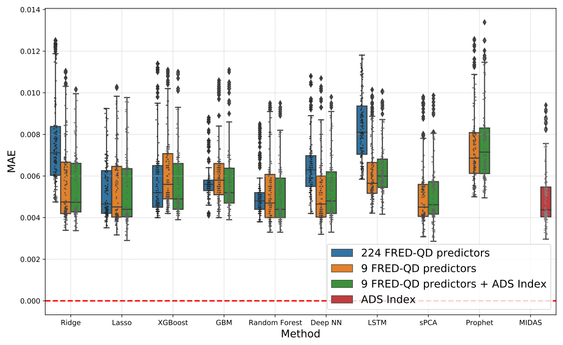
(a) one-period ahead forecasts



(b) two-period ahead forecasts



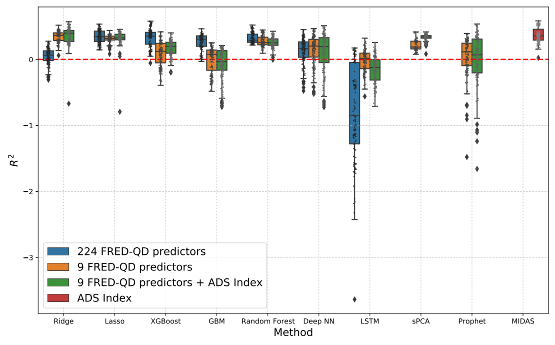
(c) three-period ahead forecasts



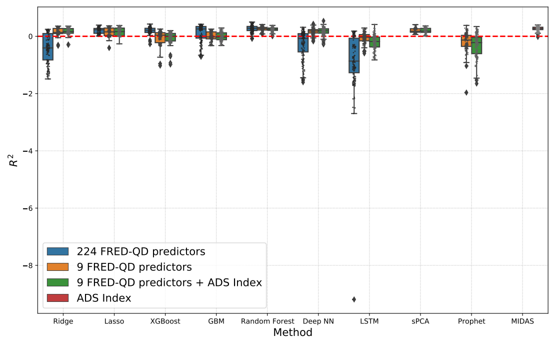
(d) four-period ahead forecasts

[*] The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.

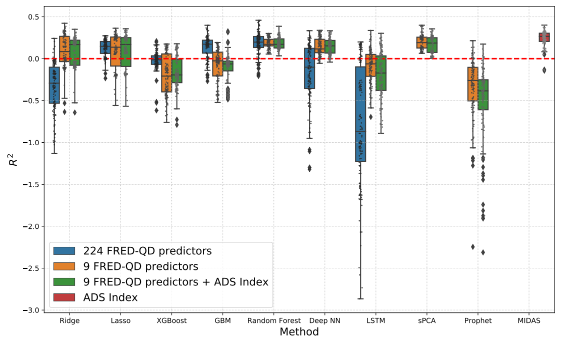
Figure 23: Box plots of OoS $R^2$ of forecasts produced by various methods across three different sets of predictors (Preds (i), Preds (ii), and Preds (iii))



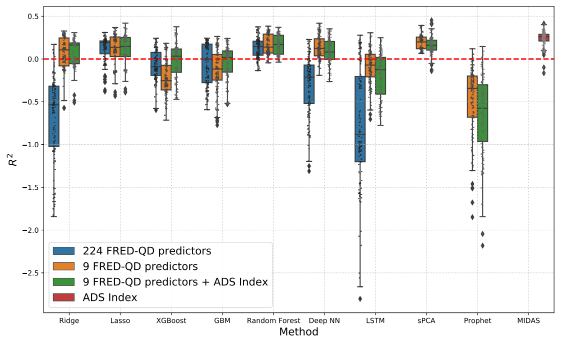
(a) one-period ahead forecasts



(b) two-period ahead forecasts



(c) three-period ahead forecasts



(d) four-period ahead forecasts

* The number of forecasts constructed for each horizon (or the number of sub-samples) is 142. The number of observations in each sub-sample is 100, and the number of observations used for training and validating a model in each rolling window is 60.