Online Appendix for

“The Anatomy of Out-of-Sample Forecasting Accuracy”

Not for Publication

July 9, 2024

A.1. Linear Prediction Model Without Interactions

Suppose that the prediction model is linear without interactions: $f(\boldsymbol{x}_t) = \alpha + \sum_{p=1}^{P} \beta_p x_{p,t}$. The fitted prediction model is given by $\hat{f}(\boldsymbol{x}_t) = \hat{\alpha} + \sum_{p=1}^{P} \hat{\beta}_p x_{p,t}$, where $\hat{\alpha}, \hat{\beta}_1, \ldots, \hat{\beta}_P$ are estimates of $\alpha, \beta_1, \ldots, \beta_P$, respectively. In this case, the Shapley value in Equation (8) in the paper is given by

$$\phi_p(\boldsymbol{x}_t\,; W_i, h) = \hat{\beta}_p(x_{p,t} - \bar{x}_p) \tag{A.1}$$

for $p \in S$ and $t \in W_i$, where $\bar{x}_p$ is the sample mean of $x_{p,t}$ for the training sample. The Shapley value in Equation (14) in the paper is given by

$$\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}\,; W_i, h\big) = \hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big) \tag{A.2}$$

for $p \in S$ and $i = 1, \ldots, D - (h-1)$, where $\hat{\beta}_p$ and $\bar{x}_p$ are again the estimate of β_p and sample mean of $x_{p,t}$, respectively, based on the training sample.

Because the loss function can be nonlinear, for a prediction model that is linear without interactions, we do not have a simple expression analogous to Equations (A.1) and (A.2) for the local PBSV_p. Nevertheless, we can derive an analytical expression for the local PBSV_p for a specific loss function in this special case. For example, consider squared error loss for the ith out-of-sample forecast:

$$L\big(y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)}, \hat{y}_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)}\big) = \big(y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)} - \hat{y}_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)}\big)^2. \tag{A.3}$$

For a linear model without interactions and Equation (A.3), the local PBSV_p can be expressed as

$$\begin{aligned}\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h, \text{SE}\big) = \underbrace{\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big)}_{\phi_p^{\text{out}}(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h)} &\big[\big(\hat{y}_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)} - y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)}\big) \\ &- \big(y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)} - \phi_\emptyset(W_i, h)\big)\big],\end{aligned} \tag{A.4}$$

where $\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h\big) = \hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big)$ is from Equation (A.2). We can view $\phi_\emptyset(W_i, h)$ in Equation (A.4) as a naïve forecast that ignores the information in the predictors and simply uses the sample mean of the target for the training sample as the prediction. For squared error loss, the local PBSV_p measures the contribution of predictor p to the squared error for the forecast that incorporates the information in the predictors relative to the squared error for the naïve forecast that ignores the information. For a linear model without interactions, Equation (A.4) says that $\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h, \text{SE}\big)$ is proportional to the error for the forecast based on the set of predictors—after adjusting for the naïve forecast error—where the factor of proportionality is given by $\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big)$ (i.e., the Shapley value for predictor p and instance $\boldsymbol{x}_{T_{\text{in}}+(i-1)}$ for a linear model). Furthermore, the sign of $\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h, \text{SE}\big)$ in Equation (A.4) depends on the signs of $\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big)$ and the term in brackets.

To gain some intuition for Equation (A.4), suppose that the forecast is perfect:

$$\hat{y}_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)} = y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)}. \tag{A.5}$$

In addition, assume that the realized target value is greater than the naïve forecast:

$$y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)} > \phi_\emptyset(W_i, h), \tag{A.6}$$

so the term in brackets in Equation (A.4) is negative. If $\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big) > 0$, then $\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h, \text{SE}\big) < 0$. In this case, predictor p contributes to the forecast being higher than the naïve forecast—since $\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big) > 0$—which is in line with the realized target value being greater than the naïve forecast; accordingly, the local PBSV_p in Equation (A.4) deems that predictor p contributes to lowering the squared error vis-à-vis the naïve forecast.

Conversely, if $\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big) < 0$, then $\phi_p^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h, \text{SE}\big) > 0$. In this case, although the forecast is perfect, the local PBSV_p deems that predictor p increases the squared error vis-à-vis the naïve forecast, as p contributes to the forecast being below the naïve forecast, while the realized target value is above the naïve forecast. A perfect forecast together with $y_{T_{\text{in}}+i:T_{\text{in}}+h+(i-1)} > \phi_\emptyset(W_i, h)$ and $\hat{\beta}_p\big(x_{p,T_{\text{in}}+(i-1)} - \bar{x}_p\big) < 0$ imply that there are one or more other predictors $q \neq p$ for which $\hat{\beta}_q\big(x_{q,T_{\text{in}}+(i-1)} - \bar{x}_q\big) > 0$ and $\phi_q^{\text{out}}\big(\boldsymbol{x}_{T_{\text{in}}+(i-1)}; W_i, h, \text{SE}\big) < 0$, as the other predictors contribute to the forecast being higher than the naïve forecast, ultimately producing the perfect forecast.

A.2. Algorithms

We created the Python package anatomy to implement the algorithms for computing the TS-Shapley-VI_p, PBSV_p, and MAS. The algorithms divide the estimation procedure into two steps: (1) evaluate the fitted models using coalitions of predictors from the sampled permuted orders and store the forecasts; (2) compute the Shapley-based metrics from the stored forecasts. After the models are evaluated in the computationally expensive first step, arbitrary combinations of models and transformations of the forecasts can be evaluated inexpensively in the second step to compute the desired metric. Algorithm 1 provides the structure for the first step. Using the results from the first step, any metric can be computed inexpensively in the second step without the need to rerun the first step.

Algorithm 1 provides the essential components for computing the TS-Shapley-VI_p and PBSV_p: the matrix of base forecasts ($\bar{\boldsymbol{Y}}$), which contains the naïve forecast (i.e., the forecast based on the empty coalition of predictors) for out-of-sample period t ($t = 1\ldots, T$) and model k ($k = 1, \ldots, K$); the matrix of model forecasts ($\hat{\boldsymbol{Y}}$), which contains the forecast for out-of-sample period t for model k with predictor p ($p = 1, \ldots, P$) excluded and included in the coalition of predictors preceding p in the mth sample $\boldsymbol{\mathcal{O}}_{m,\cdot} \in \pi(P)$, denoted by $\hat{\boldsymbol{Y}}(t, k, p, m, 1)$ and $\hat{\boldsymbol{Y}}(t, k, p, m, 2)$, respectively. An estimate of the period-t Shapley value for predictor p for the combination forecast of the K models

Result: $\hat{\boldsymbol{Y}}$: $T \times K \times P \times 2M \times 2$ array of forecasts for T out-of-sample periods and K models evaluated over coalitions of P predictors deactivated and activated in M forward and reversed permuted orders; $\bar{\boldsymbol{Y}}$: $T \times K$ matrix of naïve forecasts (i.e., model evaluations with empty predictor coalitions)

Input: $\hat{\boldsymbol{F}}$: $T \times K$ matrix of forecast functions; $\boldsymbol{X}$: T training data matrices of sizes $\mathcal{T}_t \times P$ for $t = 1, \ldots, T$; $\boldsymbol{\mathcal{X}}$: $P \times T$ out-of-sample data matrix; M: number of ordered permutations to draw from $\pi(P)$

```
Generate permutation matrix 𝓞 of size M × P containing M permutations of {1,...,P}
for t = 1 to T do                                            // loop over out-of-sample periods
    for k = 1 to K do                                        // loop over models
        Store forecast with all predictors deactivated (naïve forecast): Ȳ_{t,k} = (1/𝒯_t) Σ_{s=1}^{𝒯_t} F̂_{t,k}(X^{(t)}_{s,·})
        for m = 1 to M do                                    // loop over permutations
            Copy order to preserve it across runs: o† = {𝓞_{m,1}, ..., 𝓞_{m,P}}
            for i ∈ {0,1} do                                 // original and reverse order
                Copy training data to preserve it across runs: X† = X^{(t)}
                Initialize previous activation as naïve forecast: ŷ_pre = Ȳ_{t,k}
                for p ∈ {o†_1, ..., o†_P} do                 // loop over predictors
                    Store forecast with previously activated predictors: Ŷ_{t,k,p,iM+m,1} = ŷ_pre
                    Activate predictor p in X† by setting all elements of column p to 𝓧_{p,t}: X†_{·,p} = 𝓧_{p,t}
                    Store forecast with p and previously activated predictors:
                      Ŷ_{t,k,p,iM+m,2} = (1/𝒯_t) Σ_{s=1}^{𝒯_t} F̂_{t,k}(X†_{s,·})
                    Update previous activation for next iteration: ŷ_pre = Ŷ_{t,k,p,iM+m,2}
                end
                Reverse o† for antithetic sampling
            end
        end
    end
end
```

Algorithm 1: Forecast evaluation of permuted orders of predictors for interventional Shapley values

in $\hat{\boldsymbol{F}}_t$ can be computed directly from the $2M$ samples contained in $\hat{\boldsymbol{Y}}(t, \cdot, p, \cdot, 1)$ and $\hat{\boldsymbol{Y}}(t, \cdot, p, \cdot, 2)$:

$$\hat{\phi}_{p,t}^{(\hat{\boldsymbol{F}}_t)} = \frac{1}{2M} \sum_{m=1}^{2M} \Bigg[\underbrace{\frac{1}{K} \sum_{k=1}^{K} \hat{\boldsymbol{Y}}(t, k, p, m, 2)}_{\hat{y}_{+p,t,m}^{(\hat{\boldsymbol{F}}_t)}} - \underbrace{\frac{1}{K} \sum_{k=1}^{K} \hat{\boldsymbol{Y}}(t, k, p, m, 1)}_{\hat{y}_{-p,t,m}^{(\hat{\boldsymbol{F}}_t)}} \Bigg] \tag{A.7}$$

for $p = 1, \ldots, P$ and $t = 1, \ldots, T$, where $\hat{y}_{+p,t,m}^{(\hat{\boldsymbol{F}}_t)}$ and $\hat{y}_{-p,t,m}^{(\hat{\boldsymbol{F}}_t)}$ denote the period-t equal-weighted combination forecasts of the K fitted models in $\hat{\boldsymbol{F}}_t$ with p included and excluded, respectively, in the coalition of predictors preceding p in the mth sample of permuted orders.[1] The base prediction and the estimated Shapley values sum to the period-t equal-weighted combination forecast of the

[1] As explained in the paper, to avoid look-ahead bias, the fitted models in $\hat{\boldsymbol{F}}_t$ are based on data through period $t - 1$.

K models in $\hat{\boldsymbol{F}}_t$:

$$\hat{y}_t^{\mathrm{EW}} = \hat{\phi}_{\emptyset,t}^{(\hat{\boldsymbol{F}}_t)} + \sum_{p=1}^{P} \hat{\phi}_{p,t}^{(\hat{\boldsymbol{F}}_t)} \tag{A.8}$$

for $t = 1, \ldots, T$, where the base contribution to the forecast is given by

$$\hat{\phi}_{\emptyset,t}^{(\hat{\boldsymbol{F}}_t)} = \frac{1}{K} \sum_{k=1}^{K} \bar{\boldsymbol{Y}}_{t,k}. \tag{A.9}$$

To compute the TS-Shapley-VI$_p$, we take the average absolute value of $\hat{\phi}_{p,t}^{(\hat{\boldsymbol{F}}_t)}$ in Equation (A.7) over the out-of-sample period: $(1/T)\sum_{t=1}^{T}\left|\hat{\phi}_{p,t}^{(\hat{\boldsymbol{F}}_t)}\right|$ for $p = 1, \ldots, P$.

We can similarly decompose the squared error to obtain the local PBSV$_p$ by wrapping the loss around the forecasts:

$$\hat{\phi}_{p,t}^{(L=\mathrm{SE})} = \frac{1}{2M} \sum_{m=1}^{2M} \left[L\left(y_t, \hat{y}_{+p,t,m}^{(\hat{\boldsymbol{F}}_t)}\right) - L\left(y_t, \hat{y}_{-p,t,m}^{(\hat{\boldsymbol{F}}_t)}\right) \right] \tag{A.10}$$

for $p = 1, \ldots, P$ and $t = 1, \ldots, T$, where $\mathrm{SE}(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$. The local PBSV$_p$ measures in Equation (A.10) sum to the squared error for the period-t equal-weighted combination forecast of the K models in $\hat{\boldsymbol{F}}_t$:

$$\mathrm{SE}\big(y_t, \hat{y}_t^{\mathrm{EW}}\big) = \hat{\phi}_{\emptyset,t}^{(L=\mathrm{SE})} + \sum_{p=1}^{P} \hat{\phi}_{p,t}^{(L=\mathrm{SE})} \tag{A.11}$$

for $t = 1, \ldots, T$, where the base contribution to the squared loss is given by

$$\hat{\phi}_{\emptyset,t}^{(L=\mathrm{SE})} = \mathrm{SE}\left(y_t, \hat{\phi}_{\emptyset,t}^{((\hat{\boldsymbol{F}}_t)}\right). \tag{A.12}$$

We can also decompose the global PBSV$_p$—based, for example, on the root mean squared error (RMSE) criterion—for the out-of-sample period ($t = 1, \ldots, T$):

$$\hat{\phi}_{p}^{(L=\mathrm{RMSE})} = \frac{1}{2M} \sum_{m=1}^{2M} \left[L\left(y_{1:T}, \hat{y}_{+p,1:T,m}^{(\hat{\boldsymbol{F}}_t)}\right) - L\left(y_{1:T}, \hat{y}_{-p,1:T,m}^{(\hat{\boldsymbol{F}}_t)}\right) \right] \tag{A.13}$$

for $p = 1, \ldots, P$, where $\text{RMSE}(y_{1:T}, \hat{y}_{1:T}) = \left[(1/T)\sum_{t=1}^{T}(y_t - \hat{y}_t)^2\right]^{0.5}$. The global PBSV_p measures in Equation (A.13) sum to the RMSE for the equal-weighted combination forecast of the K models in $\hat{\boldsymbol{F}}_t$ for the out-of-sample period:

$$\text{RMSE}\big(y_{1:T}, \hat{y}_{1:T}^{\text{EW}}\big) = \hat{\phi}_{\emptyset}^{(L\,=\,\text{RMSE})} + \sum_{p=1}^{P} \hat{\phi}_{p}^{(L\,=\,\text{RMSE})}, \tag{A.14}$$

where the base contribution to the RMSE is given by

$$\hat{\phi}_{\emptyset}^{(L\,=\,\text{RMSE})} = \text{RMSE}\left(y_{1:T}, \hat{\phi}_{\emptyset,1:T}^{(\hat{\boldsymbol{F}}_t)}\right). \tag{A.15}$$

A.3. Forecasting Model Details

A natural starting point for generating an inflation forecast based on $\boldsymbol{x}_t$ is a linear predictive regression:

$$\pi_{t+1:t+h} = \underbrace{\alpha + \boldsymbol{x}_t'\boldsymbol{\beta}}_{f(\boldsymbol{x}_t)} + \varepsilon_{t+1:t+h}, \tag{A.16}$$

where α is the intercept, and $\boldsymbol{\beta} = [\ \beta_1 \quad \cdots \quad \beta_P\ ]'$ is a P-dimensional vector of slope coefficients. It is straightforward to estimate Equation (A.16) via ordinary least squares (OLS), leading to the forecast:

$$\hat{\pi}_{t+1:t+h}^{\text{OLS}} = \hat{\alpha}^{\text{OLS}} + \boldsymbol{x}_t'\hat{\boldsymbol{\beta}}^{\text{OLS}}, \tag{A.17}$$

where $\hat{\alpha}^{\text{OLS}}$ and $\hat{\boldsymbol{\beta}}^{\text{OLS}}$ are the OLS estimates of α and $\boldsymbol{\beta}$, respectively, in Equation (A.16) based on data through t. Although straightforward to compute, the forecast in Equation (A.17) tends to perform poorly in practice. By construction, OLS maximizes the fit of the model over the training sample, which can result in in-sample overfitting and thus poor out-of-sample performance, especially since the signal-to-noise ratio for inflation is limited.

A.3.1. Principal Component Regression

An ample literature employs principal component regression (PCR, Stock and Watson, 2002a,b) as a dimension-reduction technique for large datasets to forecast macroeconomic variables, including inflation (e.g., Stock and Watson, 1999; Bernanke and Boivin, 2003; Banerjee and Marcellino, 2006). Let $\boldsymbol{z}_t = [\begin{array}{ccc} z_{1,t} & \cdots & z_{C,t} \end{array}]'$ denote the vector containing the first C principal components corresponding to $\boldsymbol{x}_t$, where $C \ll P$. The PCR specification can be expressed as

$$\pi_{t+1:t+h} = \alpha_z + \boldsymbol{z}_t' \boldsymbol{\beta}_z + \varepsilon_{t+1:t+h}, \tag{A.18}$$

where $\boldsymbol{\beta}_z = [\begin{array}{ccc} \beta_{z,1} & \cdots & \beta_{z,C} \end{array}]'$ is a C-dimensional vector of slope coefficients. The forecast corresponding to Equation (A.18) is given by

$$\hat{\pi}_{t+1:t+h}^{\text{PCR}} = \hat{\alpha}_z^{\text{OLS}} + \hat{\boldsymbol{z}}_t' \hat{\boldsymbol{\beta}}_z^{\text{OLS}}, \tag{A.19}$$

where $\hat{\alpha}_z^{\text{OLS}}$ and $\hat{\boldsymbol{\beta}}_z^{\text{OLS}}$ are the OLS estimates of α_z and $\boldsymbol{\beta}_z$, respectively, in Equation (A.18), and $\hat{\boldsymbol{z}}_t$ is the C-dimensional vector of the first C principal components computed from $\boldsymbol{x}_t$, all of which are based on data through t. Because the principal components are linear combinations of the underlying predictors in $\boldsymbol{x}_t$, the PCR forecast itself is linear in the predictors. Intuitively, we extract a limited set of principal components from $\boldsymbol{x}_t$ to estimate the key latent variables that underlie the comovements among the entire set of predictors, which filters some of the noise from the predictors. The principal components then serve as predictors in a low-dimensional predictive regression with uncorrelated explanatory variables.[2] We select L in $\boldsymbol{\pi}_{t-L:t}^{\text{AR}}$ and C by choosing the combination that maximizes the adjusted R^2 for the training sample (allowing for maximum values of eleven and ten for L and C, respectively).

A.3.2. Elastic Net

Next, we use the elastic net (ENet, Zou and Hastie, 2005) to estimate the linear predictive regression in Equation (A.16). The ENet employs penalized regression to shrink the estimated slope coefficients toward zero to guard against overfitting, and there is evidence that penalized regression

[2] The principal components are uncorrelated by construction. Following convention, we standardize the predictors (using data through t) before computing the principal components.

helps to improve inflation forecasts (e.g., Medeiros and Mendes, 2016; Smeekes and Wijler, 2018). The ENet is a refinement of the least absolute shrinkage and selection operator (LASSO, Tibshirani, 1996), a seminal machine-learning device for implementing shrinkage. The LASSO relies on the ℓ_1 norm in its penalty term, so it can shrink slope coefficients to exactly zero, thereby performing variable selection. A potential drawback to the LASSO is that it tends to arbitrarily select a single predictor from a group of highly correlated predictors. The ENet mitigates this tendency by including both ℓ_1 and ℓ_2 components in its penalty term; the latter is from ridge regression (Hoerl and Kennard, 1970).

The objective function for ENet estimation of Equation (A.16) can be expressed as

$$\arg\min_{\alpha,\boldsymbol{\beta}} \frac{1}{2[t-(h-1)-1]}\left\{\sum_{s=1}^{t-(h-1)-1}\left[\pi_{s+1:s+h}-\left(\alpha+\boldsymbol{x}_s'\boldsymbol{\beta}\right)\right]^2\right\}+\lambda P_\delta(\boldsymbol{\beta}), \tag{A.20}$$

where

$$P_\delta(\boldsymbol{\beta}) = 0.5(1-\delta)\|\boldsymbol{\beta}\|_2^2+\delta\|\boldsymbol{\beta}\|_1; \tag{A.21}$$

$\lambda \geq 0$ is a hyperparameter that governs the degree of shrinkage; $\|\cdot\|_1$ and $\|\cdot\|_2$ are the ℓ_1 and ℓ_2 norms, respectively; and $0 \leq \delta \leq 1$ is a hyperparameter for blending the ℓ_1 and ℓ_2 components in the penalty term.[3] We follow the recommendation of Hastie et al. (2023) and set $\delta = 0.5$, which they point out results in a stronger tendency to select highly correlated predictors as a group. To tune λ, we use a walk-forward cross-validation procedure designed for a time-series context. The ENet forecast based on Equation (A.16) is given by

$$\hat{\pi}_{t+1:t+h}^{\mathrm{ENet}} = \hat{\alpha}^{\mathrm{ENet}}+\boldsymbol{x}_t'\hat{\boldsymbol{\beta}}^{\mathrm{ENet}}, \tag{A.22}$$

where $\hat{\alpha}^{\mathrm{ENet}}$ and $\hat{\boldsymbol{\beta}}^{\mathrm{ENet}}$ are the ENet estimates of α and $\boldsymbol{\beta}$, respectively, in Equation (A.16) based on data through t.

[3] The ENet objective function in Equation (A.20) reduces to that for OLS when $\lambda = 0$. If $\delta = 1$ ($\delta = 0$), then Equation (A.20) corresponds to the LASSO (ridge) objective function.

A.3.3. Random Forest

Random forests (Breiman, 2001) build on regression trees, machine-learning devices for incorporating nonlinearities in a flexible manner via multi-way interactions and higher-order effects of the predictors. A random forest uses an ensemble of "deep" decision trees and has a strong track record in macroeconomic forecasting (e.g., Medeiros et al., 2021; Borup and Schütte, 2022; Goulet Coulombe et al., 2022). A regression tree is constructed by sequentially splitting the predictor space into regions, with the final set of regions referred to as "terminal nodes" or "leaves." The prediction is the average value of the target in a given leaf. We can express the forecast corresponding to a regression tree with U leaves as

$$\hat{\pi}^{\mathrm{RT}}_{t+1:t+h} = \sum_{u=1}^{U} \bar{\pi}_u \mathbf{1}_u(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_u), \tag{A.23}$$

where the indicator function $\mathbf{1}_u(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_u) = 1$ if $\boldsymbol{x}_t \in R_u(\hat{\boldsymbol{\eta}}_u)$ for the uth region denoted by R_u (which is determined by the estimated parameter vector $\hat{\boldsymbol{\eta}}_u$) and 0 otherwise, and $\bar{\pi}_u$ is the average value of the target observations in R_u for the training sample based on data through t.

A large (i.e., deep) regression tree is typically able to capture complex nonlinear relationships in the data. However, in light of the bias-variance trade-off, it is susceptible to overfitting due to the high variance of the tree. A random forest reduces the variance by averaging forecasts across many deep regression trees, where each tree is constructed based on a bootstrap sample of the original data using a randomly selected subset of the predictors for each split. By using a randomly selected subset of the predictors, we "decorrelate" the trees to further reduce the variance. Indexing the bootstrap samples by b, the random forest forecast is given by

$$\hat{\pi}^{\mathrm{RF}}_{t+1:t+h} = \frac{1}{B} \sum_{b=1}^{B} \left[\sum_{u=1}^{U} \bar{\pi}^{(b)}_u \mathbf{1}^{(b)}_u(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_u) \right], \tag{A.24}$$

where B is the number of bootstrap samples, and $\bar{\pi}^{(b)}_u$ and $\mathbf{1}^{(b)}_u(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_u)$ are the counterparts to $\bar{\pi}_u$ and $\mathbf{1}_u(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_u)$, respectively, in Equation (A.23) for the bth bootstrap sample. We set $B = 500$ and let each tree grow fully deep. The proportion of predictors randomly selected for each split is tuned via a walk-forward cross-validation procedure.

A.3.4. XGBoost

Another strategy for forecasting with a regression tree is a boosted tree, which is based on gradient boosting (Breiman, 1997; Friedman, 2001), a sequential ensemble method for improving out-of-sample prediction. The basic idea is to fit a prediction function additively:

$$\hat{f}(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}) = \sum_{j=1}^{J} \hat{f}_j(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_j). \tag{A.25}$$

Each function $\hat{f}_j(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_j)$ on the right-hand-side of Equation (A.25) is a "weak" learner (i.e., a relatively simple model); for a boosted tree, $\hat{f}_j(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_j)$ corresponds to a fitted tree with a forecast of the form in Equation (A.23). Relatively simple models help to guard against overfitting; however, they are more likely to be exhibit substantive bias and thus poor fit. Boosting improves the fit by adding another tree that is trained using the residuals from the previous function in the sequence. In sum, boosting entails constructing a sequence of relatively "shallow" trees, which are then combined into an ensemble. While a random forest starts with a deep tree with low bias and uses bagging across a large number of trees to reduce the variance, a boosted tree starts with a shallow tree with low variance and refines the tree to reduce the bias.

Friedman (2002) proposes stochastic gradient boosting to make boosting more robust. Instead of basing each $\hat{f}_j(\boldsymbol{x}_t\,; \hat{\boldsymbol{\eta}}_j)$ in Equation (A.25) on all of the training data, each element is based on a randomly drawn (without replacement) subsample of the data. We fit boosted trees via stochastic gradient boosting using the popular XGBoost algorithm (Chen and Guestrin, 2016), where we tune the hyperparameters for the algorithm using a walk-forward cross-validation procedure. XGBoost performs well in forecasting competitions in a range of domains.

A.3.5. Neural Network

Our final forecasting model is a feedforward neural network. Neural networks are flexible machine-learning devices that permit general forms of nonlinearities. A neural network contains multiple layers. The first is the input layer, which is comprised of the set of predictors, followed by $L \geq 1$ hidden layers. Each hidden layer l contains P_l neurons, where each neuron takes signals from the neurons in the previous layer to generate a subsequent signal via a nonlinear activation function:

$$h_m^{(l)} = g\left(\omega_{m,0}^{(l)} + \sum_{j=1}^{P_{l-1}} \omega_{m,j}^{(l)} h_j^{(l-1)}\right) \tag{A.26}$$

for $m = 1, \ldots, P_l$ and $l = 1, \ldots, L$, where $h_m^{(l)}$ is the signal corresponding to the mth neuron in the lth hidden layer[4]; $\omega_{m,0}^{(l)}, \omega_{m,1}^{(l)}, \ldots, \omega_{m,P_{l-1}}^{(l)}$ are weights; and $g(\cdot)$ is the activation function. The output layer is the final layer. It takes the signals from the last hidden layer and converts them into a prediction:

$$\hat{\pi}_{t+1:t+h}^{\text{NN}} = \omega_0^{(L+1)} + \sum_{j=1}^{P_L} \omega_j^{(L+1)} h_j^{(L)}. \tag{A.27}$$

The activation function determines the strength of the signal passed through the network. For the activation function, we use the popular rectified linear unit (ReLU) function: $g(x) = \max\{x, 0\}$. The interactions in the network and activation function permit complex nonlinearities as the inputs feed through to the hidden layers and finally to the output layer.

Theoretically, a single hidden layer is sufficient for approximating any smooth function (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989; Hornik, 1991; Barron, 1994); however, there are potential advantages to including multiple hidden layers in neural networks (Goodfellow et al., 2016; Rolnick and Tegmark, 2018). Determining the neural network architecture (i.e., the number of hidden layers and the number of neurons in each layer) for a given application is largely an empirical matter, and we cannot know that the optimal architecture has been selected (Goodfellow et al., 2016). Accordingly, we choose an equal-weighted ensemble of two different architectures: a "shallow" neural network with one hidden layer and a "deep" neural network with three hidden layers. We follow a conventional geometric pyramid rule (Masters, 1993) in setting the number of neurons in the hidden layers, so the shallow neural network has $\lceil\sqrt{P}\rceil$ neurons in its hidden layer, while the deep neural network has $\lceil P^{3/4}\rceil$, $\lceil P^{2/4}\rceil$, and $\lceil P^{1/4}\rceil$ in its first, second, and third hidden layers, respectively.

We fit the neural networks (i.e., estimate the weights) by minimizing the training sample MSE using the Adam stochastic gradient descent algorithm (Kingma and Ba, 2015). To reduce the

[4]For the first hidden layer, $h_j^{(0)} = x_{j,t}$ for $j = 1, \ldots, P$.

influence of the random number generator in the initialization of the weights when fitting the neural networks, we fit each model 199 times with a different seed each time and use the median of the predictions.[5]

A.4. Inflation Predictors

The data for the inflation predictors are from two sources. The first is the FRED-MD database (McCracken and Ng, 2016). Table A.1 lists the 118 variables from FRED-MD and their abbreviations. The second source is the University of Michigan Survey of Consumers, from which we use three variables: Index of Consumer Sentiment (`soc_ics`), Index of Consumer Expectations (`soc_ice`), and Index of Current Economic Conditions (`soc_icc`). The variables from the University of Michigan Survey of Consumers are specified in levels.

References

Banerjee, A. and M. Marcellino (2006). Are there any reliable leading indicators for US inflation and GDP growth? *International Journal of Forecasting 22*(1), 137–151.

Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning 14*(1), 115–133.

Bernanke, B. S. and J. Boivin (2003). Monetary policy in a data-rich environment. *Journal of Monetary Economics 50*(3), 525–546.

Borup, D. and E. C. M. Schütte (2022). In search of a job: Forecasting employment growth using Google Trends. *Journal of Business & Economic Statistics 40*(1), 186–200.

Breiman, L. (1997). Arcing the edge. Technical Report 486, Statistics Department, University of California, Berkeley.

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

[5] Although the Adam algorithm is a powerful optimizer, it is our experience that neural networks at times get stuck near local minima. Using the median of 199 fitted neural networks substantially reduces the influence of local minima in computing the prediction. We fit the neural networks using the scikit-learn package in Python. We augment the objective function with an ℓ_2 penalty term and set the hyperparameter for the ℓ_2 penalty term to 0.0001 in the MPLregressor function. The batch size and number of epochs are set to 32 and 1,000, respectively.

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.

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

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

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis 38*(4), 367–378.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks 2*(3), 183–192.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning.* Boston: MIT Press.

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

Hastie, T., J. Qian, and K. Tay (2023). An introduction to `glmnet`. Manuscript.

Hoerl, A. E. and R. W. Kennard (1970). Ridge regression: Applications to nonorthogonal problems. *Technometrics 12*(1), 69–82.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks 4*(2), 251–257.

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

Kingma, D. P. and J. Ba (2015). Adam: A method for stochastic optimization. In *Proceedings of the Third Annual International Conference on Learning Representations.*

Masters, T. (1993). *Practical Neural Network Recipes in C++.* Boston: Academic Press.

McCracken, M. W. and S. Ng (2016). FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics 34*(4), 574–589.

Medeiros, M. C. and E. F. Mendes (2016). ℓ_1-regularization of high-dimensional time-series models with non-Gaussian and heteroskedastic errors. *Journal of Econometrics 191*(1), 255–271.

Medeiros, M. C., G. F. R. Vasconcelos, Álvaro 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.

Rolnick, D. and M. Tegmark (2018). The power of deeper networks for expressing natural functions. In *Sixth Annual International Conference on Learning Representations.*

Smeekes, S. and E. Wijler (2018). Macroeconomic forecasting using penalized regression methods. *International Journal of Forecasting 34*(3), 408–430.

Stock, J. H. and M. W. Watson (1999). Forecasting inflation. *Journal of Monetary Economics 44*(2), 293–335.

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.

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

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

Table A.1. FRED-MD variables

The table lists the 118 variables and their abbreviations from FRED-MD that are used as inflation predictors. The third and sixth columns indicate the transformations for the predictors.

(1) Variable	(2) Abbreviation	(3) Transformation	(4) Variable	(5) Abbreviation	(6) Transformation
Real Personal Income	rpi	Δ log	All Employees: Total Nonfarm	payems	Δ log
Real Personal Income Excluding Current Transfer Receipts	w875rx1	Δ log	All Employees: Goods-Producing Industries	usgood	Δ log
Real Consumption	dpcera3m086sbea	Δ log	All Employees: Mining	ces1021000001	Δ log
Real Manufacturing & Trade Industries Sales	cmrmtsplx	Δ log	All Employees: Construction	uscons	Δ log
Retail & Food Services Sales	retailx	Δ log	All Employees: Manufacturing	manemp	Δ log
Industrial Production: Total Index	indpro	Δ log	All Employees: Durable Goods	dmanemp	Δ log
Industrial Production: Final Products and Nonindustrial Supplies	ipfpnss	Δ log	All Employees: Nondurable Goods	ndmanemp	Δ log
Industrial Production: Final Products	ipfinal	Δ log	All Employees: Service-Providing Industries	srvprd	Δ log
Industrial Production: Consumer Goods	ipcongd	Δ log	All Employees: Trade, Transportation & Utilities	ustpu	Δ log
Industrial Production: Durable Consumer Goods	ipdcongd	Δ log	All Employees: Wholesale Trade	uswtrade	Δ log
Industrial Production: Nondurable Consumer Goods	ipncongd	Δ log	All Employees: Retail Trade	ustrade	Δ log
Industrial Production: Business Equipment	ipbuseq	Δ log	All Employees: Financial Activities	usfire	Δ log
Industrial Production: Materials	ipmat	Δ log	All Employees: Government	usgovt	Δ log
Industrial Production: Durable Materials	ipdmat	Δ log	Average Weekly Hours: Goods Producing	ces0600000007	None
Industrial Production: Nondurable Materials	ipnmat	Δ log	Average Weekly Overtime Hours: Manufacturing	awotman	Δ
Industrial Production: Manufacturing (SIC)	ipmansics	Δ log	Average Weekly Hours: Manufacturing	awhman	None
Industrial Production: Residential Utilities	ipb51222s	Δ log	Housing Starts: Total	houst	log
Industrial Production: Fuels	ipfuels	Δ log	Housing Starts: Northeast	houstne	log
Capacity Utilization: Manufacturing	cumfns	Δ	Housing Starts: Midwest	houstmw	log
Non-farm vacancies	hwi	Δ log	Housing Starts: South	housts	log
HWI/(# unemployed)	hwiuratio	Δ	Housing Starts: West	houstw	log
Civilian Labor Force	clf16ov	Δ log	New Private Housing Permits: Total	permit	log
Civilian Employment	ce16ov	Δ log	New Private Housing Permits: Northeast	permitne	log
Unemployment Rate	unrate	Δ	New Private Housing Permits: Midwest	permitmw	log
Average Duration of Unemployment (Weeks)	uempmean	Δ	New Private Housing Permits: South	permits	log log
Number Unemployed—Less Than 5 Weeks	uemplt5	Δ log	New Private Housing Permits: West	permitw	log
Number Unemployed for 5–14 Weeks	uemp5to14	Δ log	New Orders for Durable Goods	amdmnox	Δ log
Number Unemployed—15 Weeks & Over	uemp15ov	Δ log	Unfilled Orders for Durable Goods	amdmuox	Δ log
Number Unemployed for 15–26 Weeks	uemp15t26	Δ log	Total Business Inventories	businvx	Δ log
Number Unemployed for 27 Weeks & Over	uemp27ov	Δ log	Total Business: Inventories to Sales Ratio	isratiox	Δ
Initial Claims	claimsx	Δ log	Monetary Base	bogmbase	Δ log

Table A.1 (continued)

(1) Variable	(2) Abbreviation	(3) Transform-ation	(4) Variable	(5) Abbreviation	(6) Transform-ation
Total Reserves of Depository Institutions	`totresns`	Δ log	Japan/US Foreign Exchange Rate	`exjpusx`	Δ log
Nonborrowed Reserves of Depository Institutions	`nonborres`	Δ log	US/UK Foreign Exchange Rate	`exusukx`	Δ log
Commercial & Industrial Loans	`busloans`	Δ log	Canada/US Foreign Exchange Rate	`excausx`	Δ log
Real Estate Loans at All Commercial Banks	`realln`	Δ log	Producer Price Index: Finished Goods	`wpsfd49207`	Δ log
Total Nonrevolving Credit	`nonrevsl`	Δ log	Producer Price Index: Finished Consumer Goods	`wpsfd49502`	Δ log
Nonrevolving Consumer Credit to Personal Income	`conspi`	Δ	Producer Price Index: Intermediate Materials	`wpsid61`	Δ log
S&P Common Stock Price Index: Composite	`s&p500`	Δ log	Producer Price Index: Crude Materials	`wpsid62`	Δ log
S&P Common Stock Price Index: Industrial	`s&p:indust`	Δ log	Crude Oil Price (Spliced WTI & Cushing)	`oilpricex`	Δ log
S&P 500 Index Dividend Yield	`s&p:divyield`	Δ	Producer Price Index: Metals & Metal Products	`ppicmm`	Δ log
S&P PE Ratio	`s&p:peratio`	Δ log	Consumer Price Index: Apparel	`cpiappsl`	Δ log
Effective Federal Funds Rate	`fedfunds`	Δ	Consumer Price Index: Transportation	`cpitrnsl`	Δ log
3-Month AA Financial Commercial Paper Rate	`cp3mx`	Δ	Consumer Price Index: Medical Care	`cpimedsl`	Δ log
3-Month Treasury Bill Rate	`tb3ms`	Δ	Consumer Price Index: Commodities	`cusr0000sac`	Δ log
6-Month Treasury Bill Rate	`tb6ms`	Δ	Consumer Price Index: Durables	`cusr0000sad`	Δ log
1-Year Treasury Note Rate	`gs1`	Δ	Consumer Price Index: Services	`cusr0000sas`	Δ log
5-Year Treasury Note Rate	`gs5`	Δ	Consumer Price Index: All Items Less Food	`cpiulfsl`	Δ log
10-Year Treasury Bond Rate	`gs10`	Δ	Consumer Price Index: All Items Less Shelter	`cusr0000sa0l2`	Δ log
Moody's Seasoned Aaa Corporate Bond Rate	`aaa`	Δ	Consumer Price Index: All Items Less Medical Care	`cusr0000sa0l5`	Δ log
Moody's Seasoned Baa Corporate Bond Rate	`baa`	Δ	Personal Consumption Expenditures Price Index: Total	`pcepi`	Δ log
3-Month Commercial Paper Minus Federal Funds Rate	`compapffx`	None	Personal Consumption Expenditures Price Index: Durable Goods	`ddurrg3m086sbea`	Δ log
3-Month Treasury Bill Minus Federal Funds Rate	`tb3smffm`	None None	Personal Consumption Expenditures Price Index: Nondurable Goods	`dndgrg3m086sbea`	Δ log
6-Month Treasury Bill Minus Federal Funds Rate	`tb6smffm`	None	Personal Consumption Expenditures Price Index: Services	`dserrg3m086sbea`	Δ log
1-Year Treasury Note Minus Federal Funds Rate	`t1yffm`	None	Average Hourly Earnings: Goods Producing	`ces0600000008`	Δ log
5-Year Treasury Note Minus Federal Funds Rate	`t5yffm`	None	Average Hourly Earnings: Construction	`ces2000000008`	Δ log
10-Year Treasury Bond Minus Federal Funds Rate	`t10yffm`	None	Average Hourly Earnings: Manufacturing	`ces3000000008`	Δ log
Moody's Aaa Corporate Bond Minus Federal Funds Rate	`aaaffm`	None	Consumer Motor Vehicle Loans Outstanding	`dtcolnvhfnm`	Δ log
Moody's Baa Corporate Bond Minus Federal Funds Rate	`baaffm`	None	Total Consumer Loans & Leases Outstanding	`dtcthfnm`	Δ log
Switzerland/US Foreign Exchange Rate	`exszusx`	Δ log	Securities in Bank Credit at All Commercial Banks	`invest`	Δ log

Figure A.1. PBSV and TS-Shapley-VI: PCR. The figure shows the PBSV_p (left axis) and TS-Shapley-VI_p (right axis) for the PCR inflation forecast for the 1990:01 to 2022:12 out-of-sample period. The predictors on the horizontal axis are the top 20 and the bottom ten ordered according to the PBSV_p in terms of improving out-of-sample forecasting accuracy. The numbers associated with the red bars are the predictor ranks according to the TS-Shapley-VI_p.

Figure A.2. PBSV and TS-Shapley-VI: ENet. See the notes to Figure A.1 with "ENet" replacing "PCR."

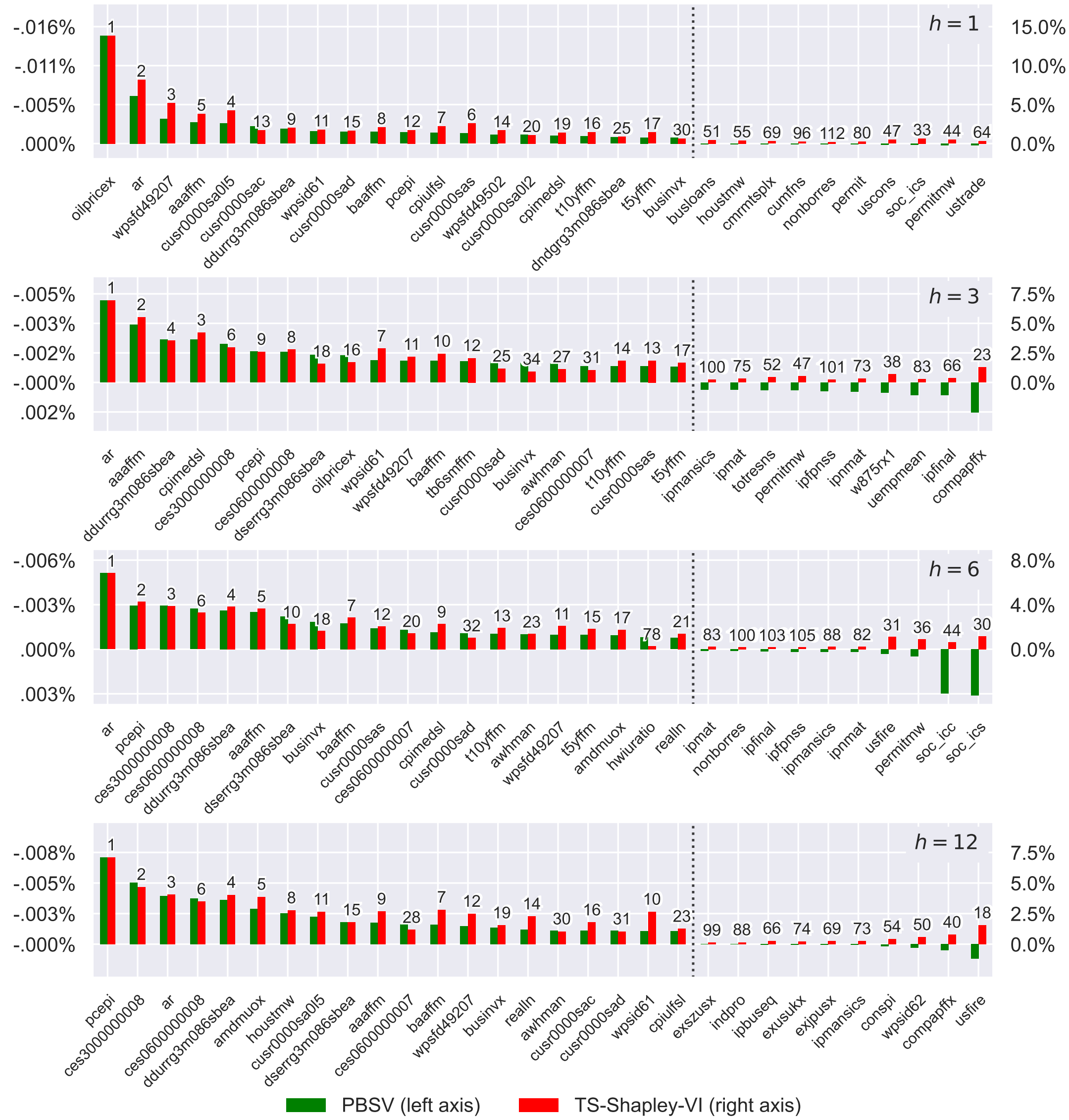

Figure A.3. PBSV and TS-Shapley-VI: random forest. See the notes to Figure A.1 with "random forest" replacing "PCR."

Figure A.4. PBSV and TS-Shapley-VI: XGBoost. See the notes to Figure A.1 with "XGBoost" replacing "PCR."

Figure A.5. PBSV and TS-Shapley-VI: neural network. See the notes to Figure A.1 with "neural network" replacing "PCR."

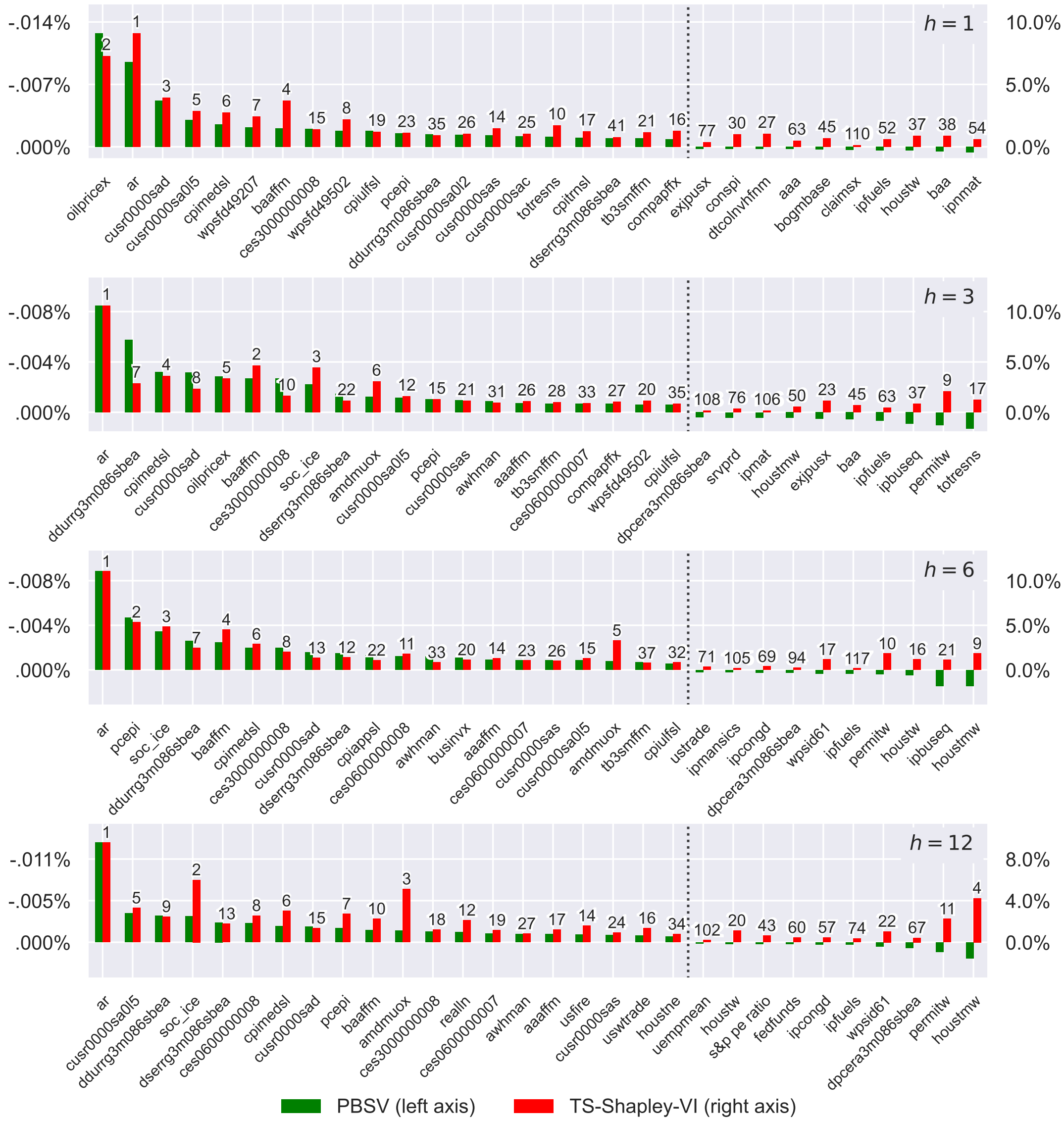

Figure A.6. PBSV and TS-Shapley-VI: ensemble-linear. See the notes to Figure A.1 with "ensemble-linear" replacing "PCR."

Figure A.7. PBSV and TS-Shapley-VI: ensemble-all. See the notes to Figure A.1 with "ensemble-all" replacing "PCR."

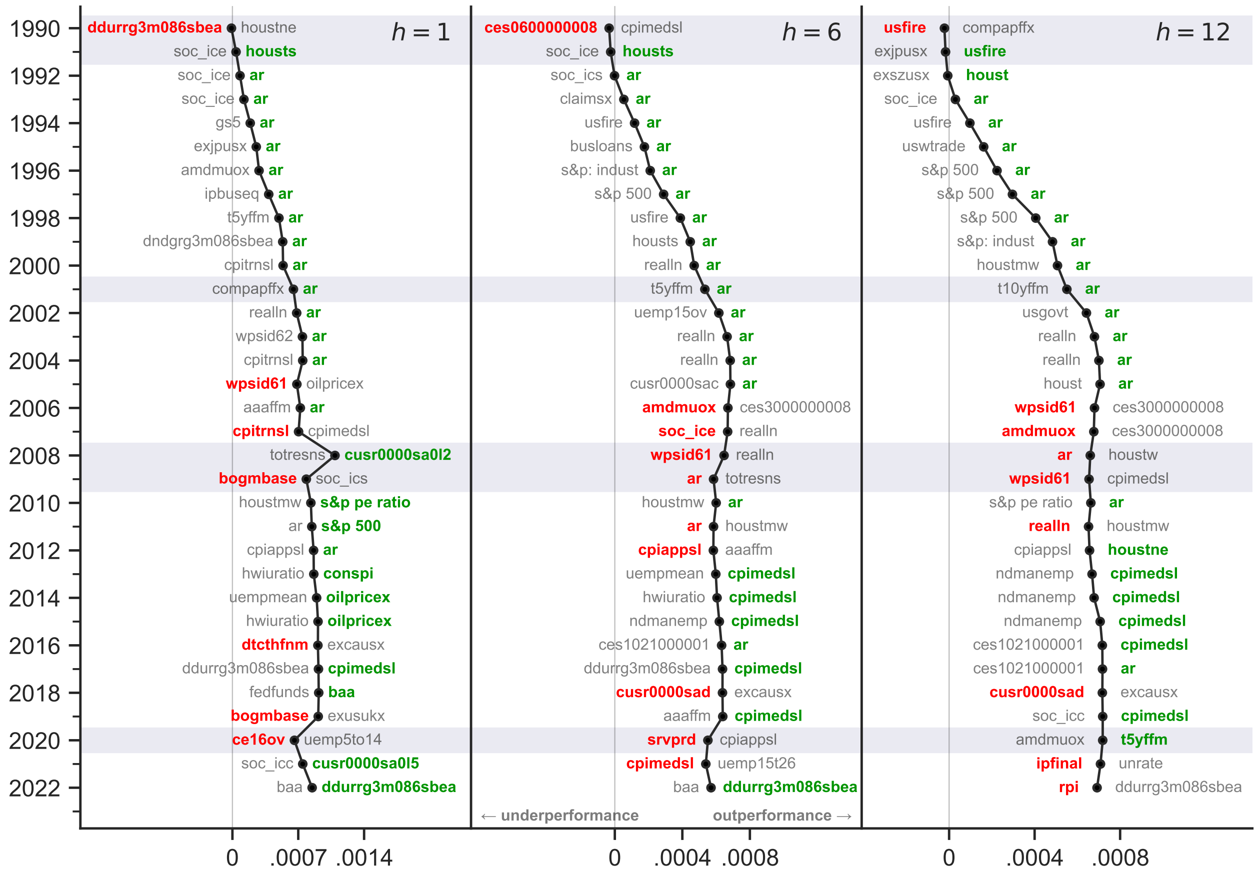

Figure A.8. Cumulative difference in squared errors: PCR. The figure shows the cumulative difference in squared errors for a naïve forecast that ignores the information in the predictors vis-à-vis the PCR forecast for the 1990:01 to 2022:12 out-of-sample period. Shifts to the right (left) imply an improvement (deterioration) in forecasting accuracy relative to the naïve forecast. The figure also shows the top (bottom) contributor to the improvement (deterioration) in forecasting performance as identified by the PBSV$_p$ for non-overlapping twelve-month subsamples; a green (red) color for the predictor indicates that the subsample is associated with an improvement (deterioration) in performance. Horizontal gray bars delineate twelve-month subsamples that contain an NBER-dated recession.

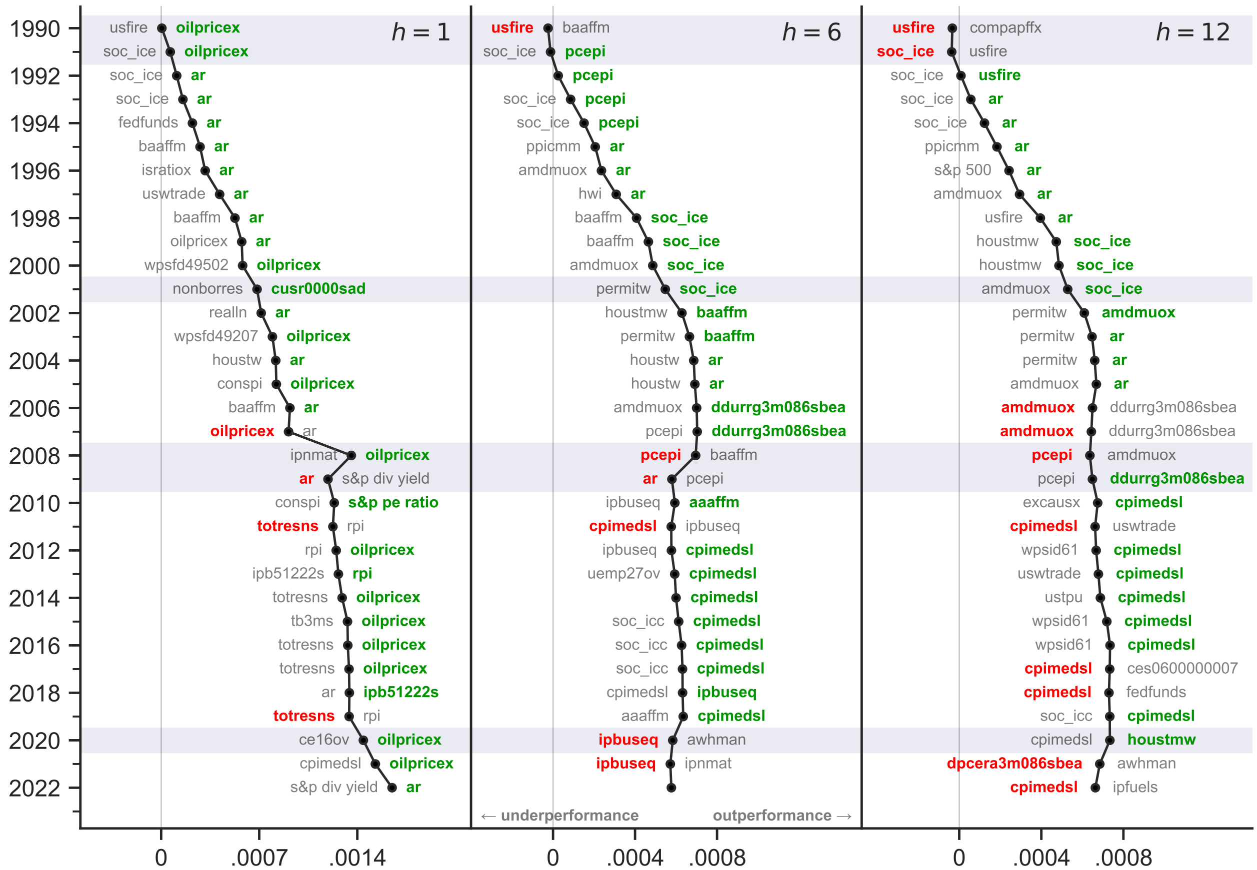

Figure A.9. Cumulative difference in squared errors: ENet. See the notes to Figure A.8 with "ENet" replacing "PCR."

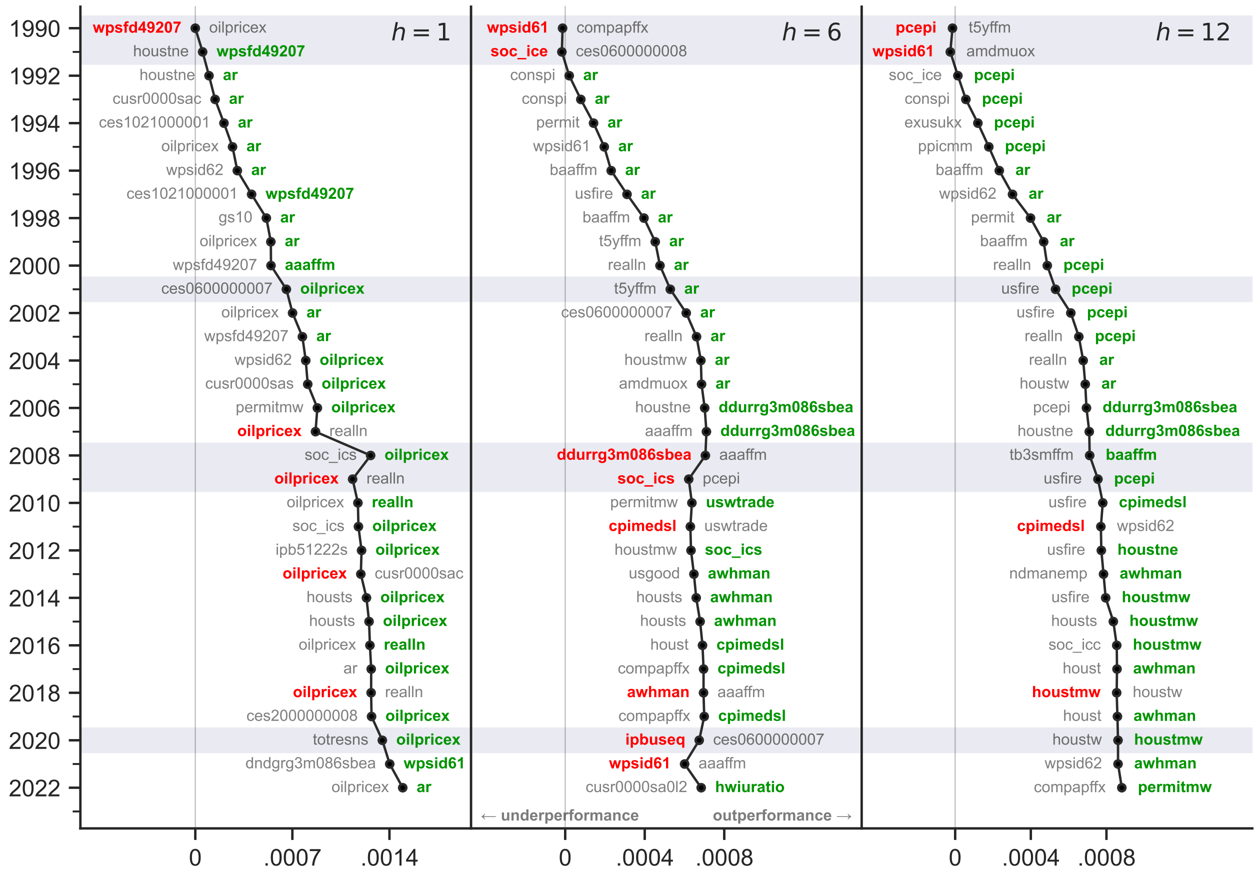

Figure A.10. Cumulative difference in squared errors: random forest. See the notes to Figure A.8 with "random forest" replacing "PCR."

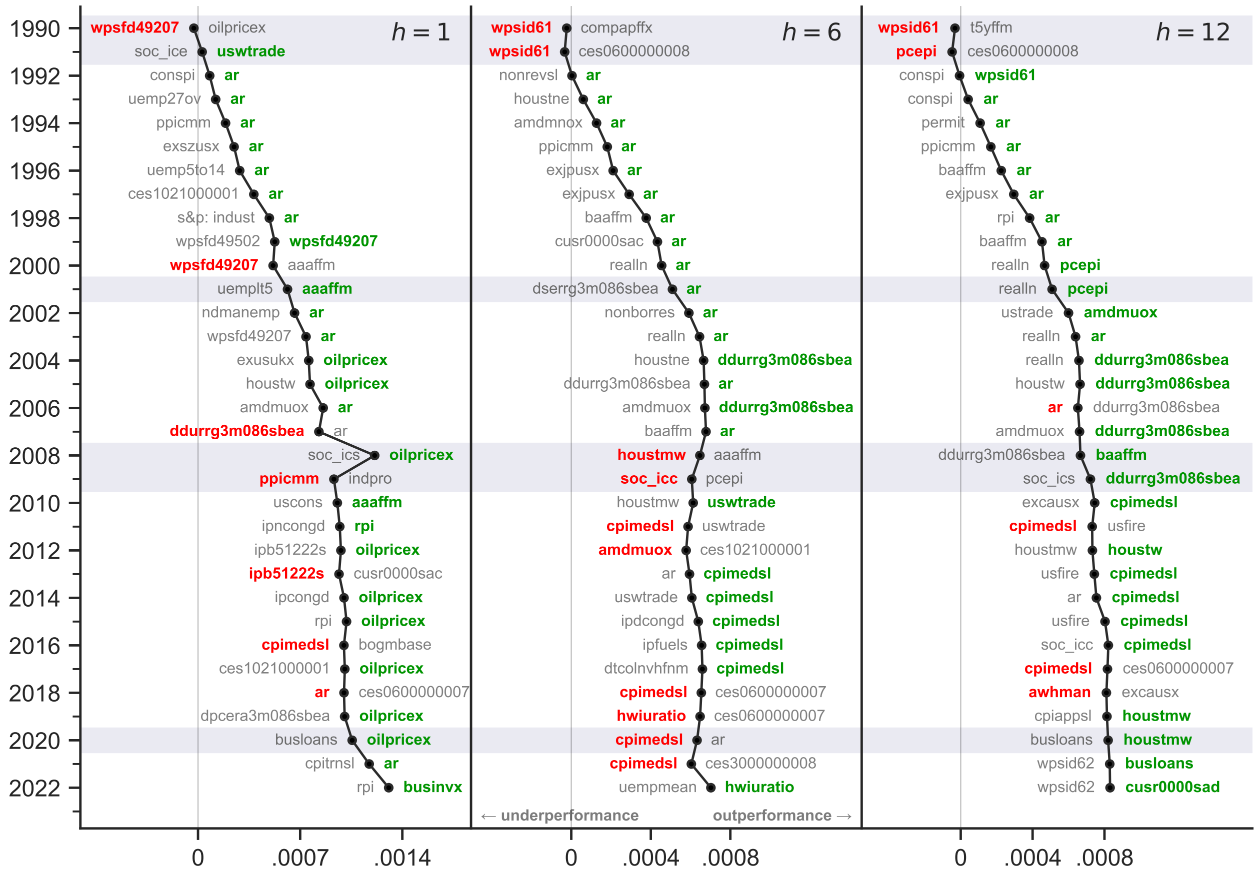

Figure A.11. Cumulative difference in squared errors: XGBoost. See the notes to Figure A.8 with "XGBoost" replacing "PCR."

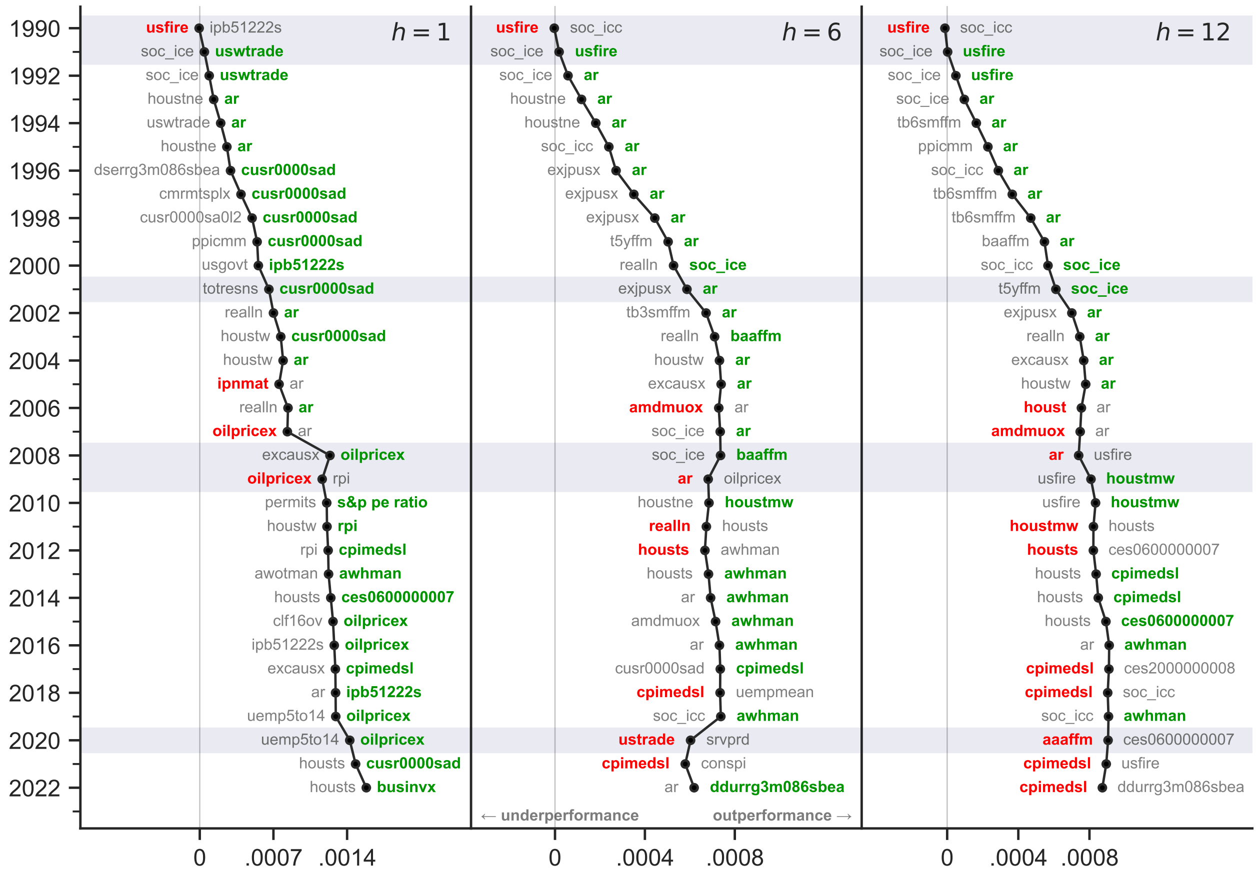

Figure A.12. Cumulative difference in squared errors: neural network. See the notes to Figure A.8 with "neural network" replacing "PCR."

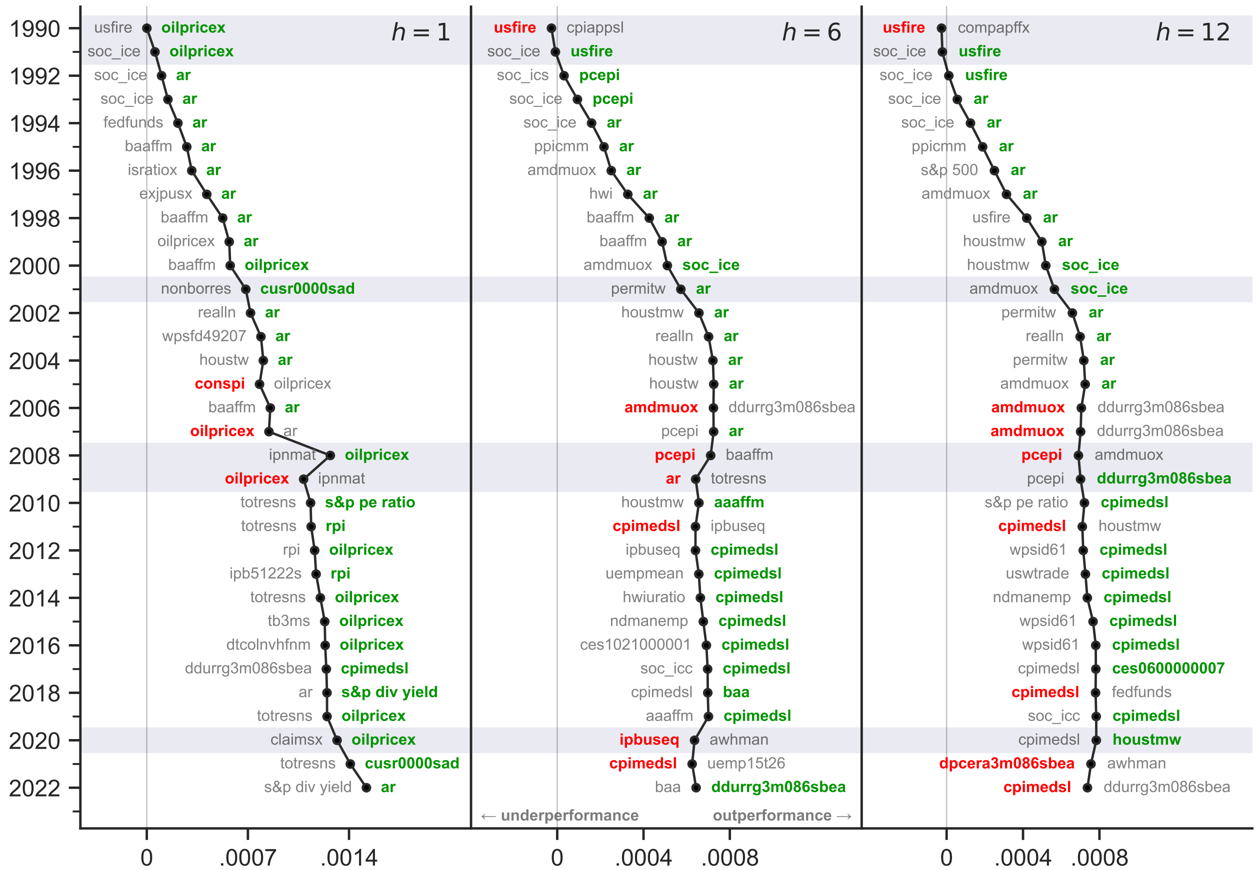

Figure A.13. Cumulative difference in squared errors: ensemble-linear. See the notes to Figure A.8 with "ensemble-linear" replacing "PCR."

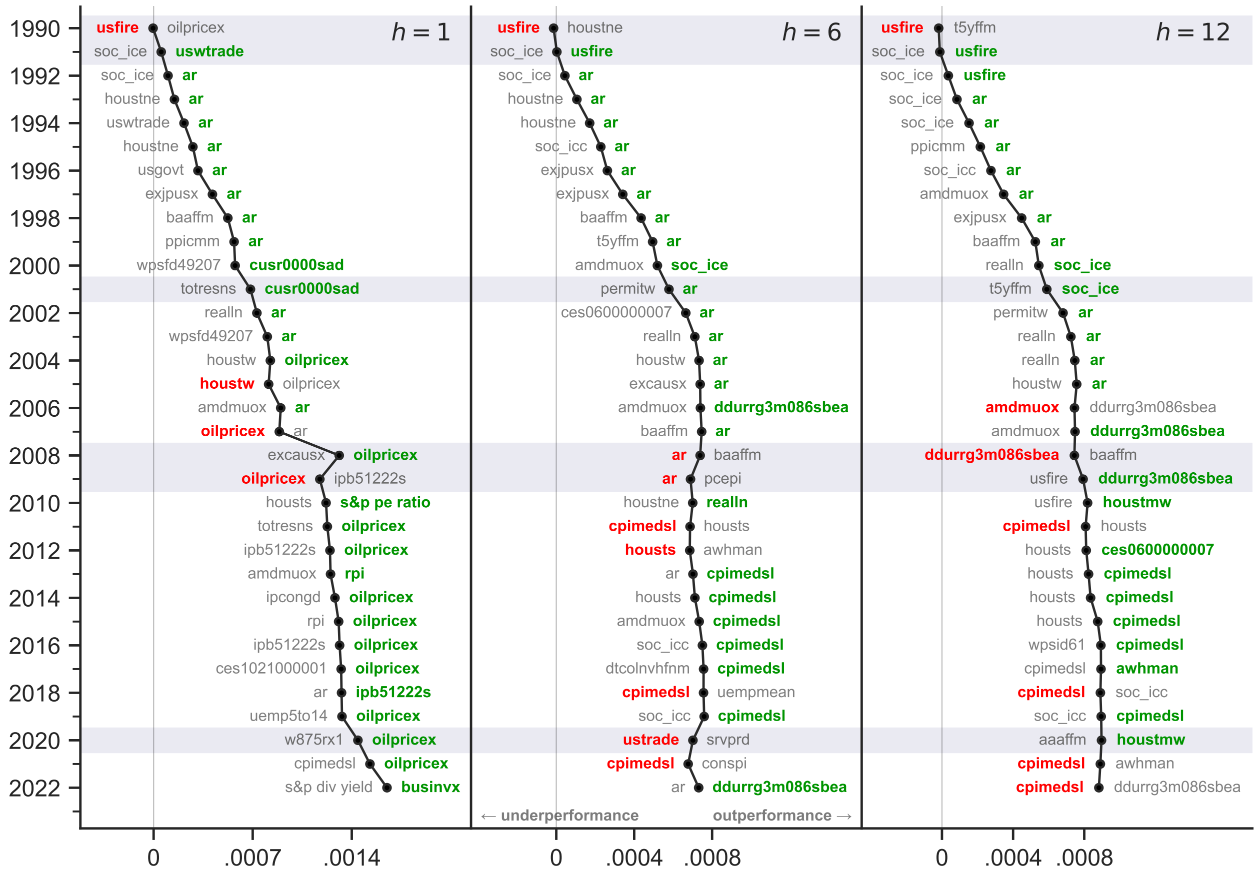

Figure A.14. Cumulative difference in squared errors: ensemble-all. See the notes to Figure A.8 with "ensemble-all" replacing "PCR."