Table 8 continued from previous page

Country	Mean	Std	Min	Max	AC1
Panama	0.158	0.435	-0.857	5.188	0.261
St. Kitts and Nevis	0.301	0.769	-1.437	4.981	0.021
St. Lucia	0.341	0.986	-2.547	4.019	-0.015
United States	0.323	0.289	-0.459	1.509	0.568
Average	0.831	1.194	-2.429	7.099	0.292
South America					
Bolivia	5.291	12.302	-1.882	103.950	0.729
Brazil	10.878	10.951	-0.511	60.098	0.915
Chile	1.193	1.024	-0.794	7.872	0.527
Colombia	1.673	0.959	-1.139	4.485	0.600
Ecuador	2.587	2.871	-0.282	18.232	-0.028
Paraguay	1.361	1.486	-5.050	9.699	0.293
Peru	7.551	13.851	-0.554	160.334	0.548
Suriname	2.770	5.129	-5.891	34.164	0.686
Trinidad and Tobago	0.673	0.857	-1.427	6.322	0.068
Uruguay	3.308	2.237	-0.582	14.334	0.556
Average	3.729	5.167	-1.811	41.949	0.489
Asia and Oceania					
Fiji	0.441	0.748	-1.272	4.670	0.129
India	0.730	0.870	-2.120	3.175	0.446
Indonesia	0.883	1.417	-1.034	11.869	0.576
Israel	3.262	4.454	-1.338	24.285	0.801
Japan	0.137	0.508	-1.004	2.024	0.118
Jordan	0.429	2.495	-13.991	13.601	-0.008
Korea, Rep.	0.510	0.726	-1.189	4.237	0.411
Malaysia	0.287	0.544	-1.954	3.724	-0.081
Myanmar	1.387	2.325	-7.020	9.682	0.273
Nepal	0.789	1.565	-3.090	4.623	0.562
Pakistan	0.669	0.798	-1.557	4.404	0.138
Philippines	0.908	1.291	-2.353	8.829	0.453
Samoa	0.636	1.869	-4.964	7.218	0.021
Singapore	0.174	0.415	-1.071	2.000	0.161
Solomon Islands	0.884	1.358	-3.714	9.108	-0.052
Sri Lanka	0.907	1.453	-3.575	5.682	0.357
Taiwan, China	0.264	1.018	-2.610	4.404	0.027
Thailand	0.398	0.617	-0.898	3.737	0.218
Turkey	4.017	2.739	-1.675	20.479	0.284

Continued on next page

Table 8 continued from previous page

Country	Mean	Std	Min	Max	AC1
Average	0.932	1.432	-2.970	7.776	0.254
		Africa			
Algeria	1.044	2.531	-5.099	10.795	-0.045
Botswana	0.846	0.594	-0.872	3.683	0.189
Burundi	0.855	2.292	-5.039	15.763	-0.044
Burkina Faso	0.326	2.471	-12.437	11.754	-0.180
Cameroon	0.551	1.831	-7.696	9.169	0.054
Cote d'Ivoire	0.481	1.622	-5.300	7.685	0.006
Egypt, Arab Rep.	1.045	2.368	-6.899	9.844	-0.186
Eswatini	0.930	2.263	-14.377	16.594	-0.319
Ethiopia	0.476	2.510	-5.790	8.794	0.088
Gambia	0.860	1.769	-3.532	11.654	0.278
Ghana	2.487	3.246	-9.856	21.063	0.700
Kenya	1.097	1.650	-3.121	9.782	0.170
Madagascar	1.320	2.133	-9.471	10.589	0.262
Malawi	1.735	3.140	-9.249	13.410	0.188
Mauritius	0.633	1.023	-2.326	4.567	0.011
Morocco	0.466	0.921	-2.053	3.555	0.288
Niger	0.278	2.935	-18.424	14.433	0.196
Nigeria	1.801	3.715	-9.531	22.314	0.126
Senegal	0.421	1.921	-3.916	11.882	0.245
Seychelles	0.248	1.958	-18.250	17.173	0.069
South Africa	0.941	0.613	-0.421	3.599	0.169
Sudan	3.563	5.283	-14.007	24.935	0.236
Tunisia	0.531	0.591	-1.690	2.564	0.364
Average	0.997	2.147	-7.363	11.548	0.125
Global average	1.183	1.870	-3.387	11.793	0.269

The table reports the mean, standard deviation (Std), minimum (Min), maximum (Max), and first-order autocorrelation (AC1) for monthly headline inflation for each of the 100 countries in the 1985-1999 sample. Inflation is computed as $\pi_t = 100 \cdot (log(P_t) - log(P_{t-1}))$, where P_t represents the headline CPI measure for each country in the table.

Table 9: Descriptive statistics 2000-2019

Country	Mean	Std	Min	Max	AC1
		Europe			
Austria	0.160	0.343	-1.109	1.190	-0.007
Belgium	0.159	0.269	-0.609	0.933	0.187
Cyprus	0.126	0.831	-2.366	2.068	0.224
Denmark	0.131	0.368	-0.613	1.239	0.119
Finland	0.121	0.314	-0.749	1.041	0.091
France	0.118	0.313	-1.006	1.007	-0.080
Germany	0.120	0.351	-1.010	1.053	-0.249
Greece	0.153	1.155	-1.964	3.260	-0.039
Hungary	0.340	0.492	-0.672	2.449	0.362
Iceland	0.372	0.528	-0.911	3.332	0.293
Ireland	0.143	0.469	-1.610	1.310	0.235
Italy	0.137	0.212	-0.581	0.623	0.132
Luxembourg	0.161	0.667	-1.410	1.888	-0.358
Malta	0.158	0.657	-2.284	3.036	-0.042
Netherlands	0.155	0.481	-1.062	1.219	0.222
Norway	0.170	0.456	-1.284	2.347	0.027
Portugal	0.151	0.557	-1.423	1.918	0.184
Slovenia	0.236	0.580	-1.383	1.593	0.291
Spain	0.170	0.566	-1.925	1.430	0.240
Sweden	0.110	0.415	-1.352	1.021	-0.044
Switzerland	0.036	0.351	-1.052	1.126	0.157
United Kingdom	0.165	0.293	-0.703	0.876	-0.063
Average	0.163	0.485	-1.231	1.635	0.085
		North America			
Bahamas	0.155	0.408	-0.773	2.753	-0.080
Barbados	0.307	0.686	-1.620	2.389	0.312
Canada	0.156	0.371	-1.043	1.148	0.193
Costa Rica	0.541	0.537	-0.912	2.116	0.547
Dominica	0.114	0.670	-4.115	2.538	-0.196
Dominican Republic	0.625	1.290	-3.333	10.663	0.600
El Salvador	0.195	0.510	-1.290	2.929	0.095
Grenada	0.159	0.470	-1.844	2.047	0.221
Guatemala	0.439	0.449	-0.867	1.874	0.374
Haiti	0.852	0.883	-3.040	5.832	0.582
Honduras	0.505	0.375	-0.394	2.087	0.414
Jamaica	0.682	0.691	-1.025	3.271	0.476
Mexico	0.363	0.353	-0.740	1.686	0.462

Continued on next page

Table 9 continued from previous page

Country	Mean	Std	Min	Max	AC1
Panama	0.190	0.368	-0.852	1.396	0.332
St. Kitts and Nevis	0.164	0.651	-2.134	3.869	0.008
St. Lucia	0.142	0.848	-2.510	4.749	-0.073
United States	0.176	0.373	-1.934	1.215	0.490
Average	0.339	0.584	-1.672	3.092	0.280
South America					
Bolivia	0.363	0.571	-2.578	2.639	0.263
Brazil	0.504	0.380	-0.230	2.976	0.640
Chile	0.262	0.388	-1.311	1.478	0.354
Colombia	0.400	0.398	-0.312	2.257	0.699
Ecuador	0.612	1.504	-0.706	13.394	0.718
Paraguay	0.494	0.842	-2.264	3.879	0.222
Peru	0.219	0.294	-0.578	1.261	0.312
Suriname	1.093	2.197	-3.226	20.123	0.454
Trinidad and Tobago	0.436	0.761	-2.027	3.595	0.134
Uruguay	0.665	0.712	-0.734	5.632	0.359
Average	0.505	0.805	-1.397	5.723	0.415
Asia and Oceania					
Fiji	0.262	0.806	-2.083	5.376	-0.005
India	0.526	0.749	-1.660	4.474	0.227
Indonesia	0.533	0.739	-0.478	8.350	0.239
Israel	0.120	0.439	-0.901	1.527	0.335
Japan	0.012	0.292	-0.808	2.031	0.198
Jordan	0.269	0.895	-3.454	5.832	0.192
Korea, Rep.	0.197	0.372	-0.748	1.290	0.260
Malaysia	0.176	0.405	-1.246	3.894	0.294
Myanmar	1.016	1.572	-2.699	8.548	0.616
Nepal	0.513	1.471	-2.322	7.949	0.177
Pakistan	0.625	0.782	-1.332	3.283	0.208
Philippines	0.337	0.395	-1.027	2.310	0.466
Samoa	0.318	1.811	-4.306	11.952	0.106
Singapore	0.129	0.496	-1.621	1.988	-0.173
Solomon Islands	0.486	1.131	-2.049	5.032	0.139
Sri Lanka	0.640	1.284	-3.519	4.498	0.291
Taiwan, China	0.080	0.714	-1.893	2.102	-0.136
Thailand	0.161	0.477	-3.015	2.199	0.348
Turkey	1.134	1.542	-1.453	10.066	0.401

Continued on next page

Table 9 continued from previous page

Country	Mean	Std	Min	Max	AC1
Average	0.397	0.862	-1.927	4.879	0.220
		Africa			
Algeria	0.317	1.004	-2.297	3.380	-0.029
Botswana	0.535	0.535	-1.445	3.175	0.271
Burundi	0.660	1.759	-3.762	6.950	-0.021
Burkina Faso	0.154	1.185	-3.411	4.433	-0.005
Cameroon	0.183	0.494	-1.475	2.168	-0.008
Cote d'Ivoire	0.182	0.734	-2.438	4.755	0.131
Egypt, Arab Rep.	0.789	1.011	-3.411	4.707	0.386
Eswatini	0.544	1.590	-11.430	18.137	-0.072
Ethiopia	0.926	1.762	-5.960	11.996	0.533
Gambia	0.523	0.576	-0.879	4.750	0.342
Ghana	1.151	1.299	-1.536	12.042	0.443
Kenya	0.709	1.318	-3.041	10.978	0.378
Madagascar	0.688	1.061	-3.765	7.436	0.634
Malawi	1.160	3.019	-3.971	11.587	0.479
Mauritius	0.351	0.697	-2.503	3.005	0.297
Morocco	0.125	0.610	-1.579	2.210	0.163
Niger	0.153	1.344	-5.542	5.434	0.151
Nigeria	0.975	1.325	-3.490	7.163	0.152
Senegal	0.131	0.909	-3.225	2.807	0.283
Seychelles	0.426	1.727	-2.424	21.617	0.246
South Africa	0.430	0.435	-1.142	1.690	0.390
Sudan	1.538	2.598	-5.989	22.193	0.306
Tunisia	0.336	0.300	-0.684	1.148	0.233
Average	0.565	1.186	-3.278	7.555	0.247
Global average	0.384	0.795	-1.994	4.530	0.227

The table reports the mean, standard deviation (Std), minimum (Min), maximum (Max), and first-order autocorrelation (AC1) for monthly headline inflation for each of the 100 countries in the 2000-2019 sample. Inflation is computed as $\pi_t = 100 \cdot (log(P_t) - log(P_{t-1}))$, where P_t represents the headline CPI measure for each country in the table.

B. Forecasting models

This section described the machine learning models utilized to forecast inflation. We fit the models each period. We follow the most common approach in the literature and select tuning parameters adaptively from the data in a validation sample. We tune the relevant hyperparameters of the models in the first training window and then subsequently every December. The cross-validation scheme for the models is described in subsection B.4 below. All models are implemented in `Python`.

B.1. Elastic Net

The Elastic Net (EN) is a machine-learning device based on penalized regression. As a linear model, it assumes that $G_{h,t}(\boldsymbol{q}_t) = \boldsymbol{\beta}_h^{'}\boldsymbol{q}_t$, where $\boldsymbol{\beta}_h^{'}$ is a vector of slope coefficients and $\boldsymbol{q}_t$ is our vector of predictors. It alleviates overfitting by augmenting the objective function for estimating the slope coefficients with L_1 and L_2 penalty terms. Specifically, it solves:

$$\widehat{\boldsymbol{\beta}}_h = \arg\min_{\beta_h} \left[\sum_{t=1}^{T-h} (\pi_{t+h}^h - \boldsymbol{\beta}_h^{'}\boldsymbol{q}_t)^2 + \lambda \left(\alpha \sum_{i=1}^{n} \beta_{h,i}^2 + (1-\alpha) \sum_{i=1}^{n} |\beta_{h,i}| \right) \right], \tag{B.1}$$

where $\lambda > 0$ and $\alpha \in [0, 1]$.

The first term of the penalty encourages highly correlated features to be averaged, while the second encourages a sparse solution for the shrunk coefficients. In this sense, it combines the two major frameworks for penalized regression, the LASSO (L_1) of Tibshirani (1996) and Ridge regression (L_2) proposed by Hoerl and Kennard (1970). α determines the weight between the L_1 lasso penalty and the L_2 ridge penalty, and λ is a tuning parameter determining the amount of shrinkage. It is easy to see that the EN is very general as $\lambda = 0$ results in OLS and $\alpha = 1$ reduces to ridge while $\alpha = 0$ reduces to lasso.

We set $\alpha = 0.5$ and select λ by cross-validation of an automatically computed equally-spaced λ grid partitioned into 100 points for validation. See the `ElasticNetCV` documenta-

tion for more information on the computation of the alpha grid.[14] We set $\alpha = 0.5$, giving equal weight to L_1 and L_2 penalty terms, which is the `sklearn` default value. We implement the Elastic Net is using `ElasticNetCV` from the `sklearn` package.

B.2. Artificial Neural Networks

We consider a feedforward Neural Network (NN) as the first nonlinear alternative. Let $x_1, ..., x_{P_0}$, denote the P_0 dimensional set of input variables. One or more hidden layers follow. Each hidden layer l contains P_l neurons, each of which takes signals from the neurons in the previous hidden layer to generate a subsequent signal:

$$h_m^{(l)} = g\left(w_{m,0}^{(l)} + \sum_{j=1}^{P_{l-1}} w_{m,j}^{(l)} h_j^{(l-1)}\right) \quad \text{for } m = 1, \ldots, P_l;\ l = 1, \ldots, L, \tag{B.2}$$

where $h_m^{(l)}$ is the signal corresponding to the mth neuron in the lth hidden layer;[15] $w_{m,0}^{(l)}, w_{m,1}^{(l)}, \ldots, w_{m,P_{l-1}}^{(l)}$ are weights; and $g(\cdot)$ is a (nonlinear) activation function. The final layer is the output layer, which translates the signals from the last hidden layer into a prediction:

$$\widehat{\pi}_{t+h}^{h} = w_{m,0}^{(L)} + \sum_{j=1}^{P_L} w_{m,j}^{(L)} h_j^{(L-1)} \tag{B.3}$$

where $\widehat{\pi}_{t+h}^{h}$ denotes the h month ahead prediction of the target variable. Throughout this paper, we use the rectified linear unit (ReLU) function, $g(y) = \max(y, 0)$, as our nonlinear activation function. In response to a sufficiently strong signal, the activation function activates a neuronal connection and relays the signal forward through the network. To illustrate the basic structure of a feedforward NN, the diagram in Figure 7 portrays a NN consisting of five inputs and three hidden layers with four, three, and two neurons, respectively. The final layer is the output layer and, thus, the model's prediction. The myriad of interactions

[14] Alternatively, one can take a look at the relevant `github` page.

[15] For the first hidden layer, $h_j^{(0)} = x_j$ for $j = 1, \ldots, P_0$.

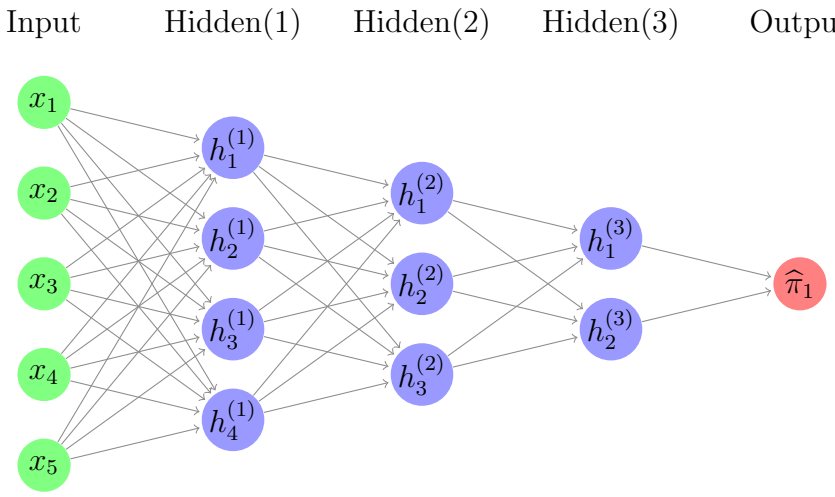


Figure 7: Graphical representation of a feedforward Neural Network

among the inputs and the neurons in the network allows for complex nonlinear relationships between the predictors.

Fitting (or training) a NN requires estimating the weights. We train the NN by minimizing the MSE between the predicted and observed values. We augment the objective function with an ℓ_1 penalty term to help guard against overfitting, which we set equal to 0.0001, the `sklearn` default value. We use the recently developed ADAM SGD algorithm (Kingma and Ba 2015) to train the NN, which we implement in `Python` using the `sklearn` package with a constant learning rate of 0.01. We use three layers in the network and 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 payer, 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. This leads to 81, 54, and 27 neurons in the first, second, and third layers, respectively. We set the number of epochs in the algorithm to 700 and the batch size to 32. To reduce the influence of the random-number generator in the initialization of the NN

estimation, we fit each model 100 times with a different seed each time and use the median of the predictions.[16]

B.3. Random Forest

The Random Forest (RF) methodology is due to Breiman (2001) who proposed to create ensembles of regression trees by bagging randomly constructed regression trees with the intent to reduce the overall variance of the model (Breiman 1996). Regression trees are highly flexible nonparametric models that approximate nonlinear functions using local predictions from a recursive partitioning of the predictor space. More specifically, a regression tree splits the predictor space into a set of regions subject to the chosen splitting predictors and the corresponding split-points and models the response by a region-specific constant value. So, regression trees split the observations into sub-groups with similar predictor values. The objective is to choose the splitting predictors and split-points such that the loss function is minimized.[17] Given our target variable, π^h_{t+h}, a vector of predictor values at time t, $\boldsymbol{q}_t$, and a partition into J regions, $R_1, R_2, ..., R_J$, the regression tree becomes:

$$\pi^h_{t+h} = \sum_{j=1}^{J} c_j \mathbf{I}(\boldsymbol{q}_t \in R_j), \tag{B.4}$$

where $\mathbf{I}$ is an indicator function and c_j is the prediction of the tree if the observation falls into region R_j. We use the sum of squared error criterion for which $\widehat{c}_j$ is simply the average of past realizations of π^h_{t+h} such that $\boldsymbol{q}_t$ belongs to R_j. We find all J regions by recursive binary splitting, which is a top-down approach meaning that all data is considered for the first split. Consider a splitting variable, m, and a split-point, s, that partitions the predictor space, Q, into two regions $R_1(m, s) = \{Q|Q_m \leq s\}$ and $R_2(m, s) = \{Q|Q_m > s\}$. We choose m and s to solve:

[16] Although the ADAM stochastic gradient descent algorithm is a very powerful optimizer, it is our experience that Neural Networks tend to get stuck in local minima. Using the median of 100 networks greatly minimizes the chance of this happening.

[17] See Hastie, Tibshirani, and Friedman (2009) for an intuitive example of a regression tree with two predictors.

$$\min_{m,s}\left[\min_{c_1}\sum_{t=1}^{T}\mathbf{I}(\boldsymbol{q}_t \in R_1(m,s))(\pi^h_{t+h}-c_1)^2+\min_{c_2}\sum_{t=1}^{T}\mathbf{I}(\boldsymbol{q}_t \in R_2(m,s))(\pi^h_{t+h}-c_2)^2\right]. \quad \text{(B.5)}$$

This procedure is typically continued iteratively on each of the resulting regions, subject to a minimum requirement for the number of observations in each region. A large regression tree is typically able to explain complex nonlinear relations in the data but tends to overfit (high variance).[18] The RF methodology reduces the variance by averaging many of such regression trees where each tree is fitted on a bootstrap sub-sample of the original data using a randomly selected subset of the predictors, indexed by $b = 1, ..., B$ where B is the total number of individual trees. The final forecast is a simple average of the B individual forecasts:

$$\widehat{\pi}^h_{t+h} = \frac{1}{B}\sum_{b=1}^{B}\left[\sum_{j=1}^{J_b}\widehat{c}_{j,b}\mathbf{I}(\boldsymbol{q}_t \in R_{j,b})\right]. \quad \text{(B.6)}$$

The proportion of predictors randomly selected for each split is chosen by cross-validation. We set $B = 500$ and the minimum leaf size to five.[19]

The number of trees considered is $B = 500$, and the trees are allowed to grow until each leaf is no smaller than five observations. With the number of potential predictors given by $p = 108$, we validate the number of predictors to be considered at each split in $[p, 2/3p, 1/3p, \sqrt{p}, \log_2 p]$. The implementation is conducted using `RandomForestRegressor` and `GridSearchCV` from the `sklearn` package.

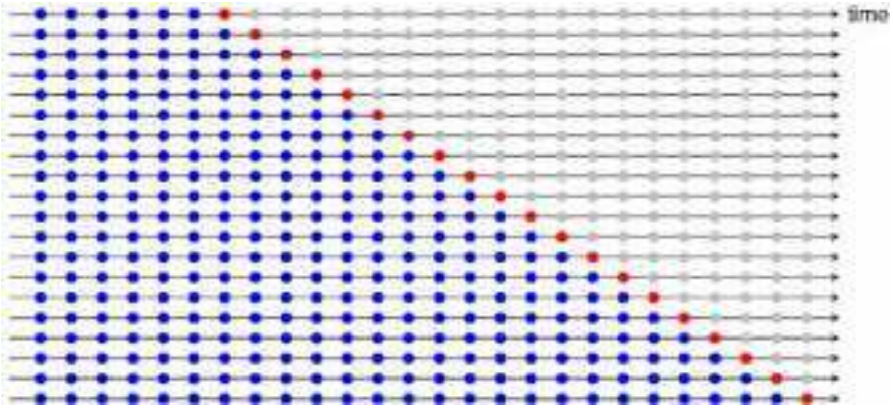
[18] One caveat of regression trees is that they cannot extrapolate to more extreme observations than those on which they are trained. This is an apparent weakness in cases where the out-of-sample values for $\boldsymbol{q}_t$ and π^h_{t+h} might be more extreme than the corresponding in-sample values. To give a concrete example in the case of inflation forecasting: if the most extreme observation in the training window is, say, $\pi^h_{t+h} = 5\%$, and the next out-of-sample value to be predicted is 10%, then, no matter how great a fit, will the regression tree have an upper bound at 5% and thus severely under-predict the target.

[19] We also tried cross-validating the minimum leaf size as well as letting the trees grow fully deep (which is consistent with Breiman (2001)), but both alternatives produced slightly weaker results.

B.4. Cross-validation

We use a walk-forward cross-validation scheme in which we preserve the time series structure of the data. For a given training window, we create 20 test-train splits where the training sets consist of observations preceding the test set. The 20 test sets are drawn to correspond to the final 20 observations in our training window, and the forecast accuracy is computed as an average across the 20 test sets. See figure 8 for an illustration of the general structure of our cross-validation scheme. We use a gap equal to the forecast horizon in question, $h-1$, between the training set and the corresponding test set to mimic the general forecasting methodology employed in the paper.

Figure 8: Illustration of walk-forward cross-validation



The figure shows an example of the walk-forward cross-validation scheme with 20 splits and a gap = 0. Blue dots form the training data, while the red dots form the test data. The figure is borrowed from Hyndman and Athanasopoulos (2018).

C. Alternative methods for factor extraction

We consider two alternative methods for factor extraction in addition to factors computed as simple averages. This section briefly outlines the main theory of PCA and Autoencoders, and displays the main results for these factor extraction methods.

C.1. PCA

The first alternative method is principal components analysis (PCA), which has a long tradition, starting with the seminal work of Stock and Watson (2002b), as the workhorse methodology in the economics and finance literature for the extraction of latent factors from high-dimensional data. PCA performs a linear projection of the data, creating principal components (low-dimensional representations) that are orthogonal to each other and ordered according to their ability to explain the variance of the data. After appropriate standardization, we compute the global factor as the first principal component from a singular value decomposition of the entire matrix of raw inflation data, $\boldsymbol{X}$. The ordering by variance ensures that the first component has the highest variance among all linear combinations of the predictors in $\boldsymbol{X}$ and thus contains the most information. The regional factors are computed as the first principal component from the subset of $\boldsymbol{X}$ relevant to the region of interest.

C.2. Autoencoder

A potential drawback of a simple average and PCA is the underlying assumption that the latent factors can be extracted as *linear* combinations of the inflation series. This assumption is challenged by past evidence in the literature of strong nonlinearities in inflation (see, for example, Binner et al. 2006, Cuestas and Harrison 2010, Medeiros et al. 2021, Coulombe et al. 2020). Since these nonlinearities in inflation have manifested in several countries, both in its time-series dimension and in its relation to other macroeconomic variables, it is

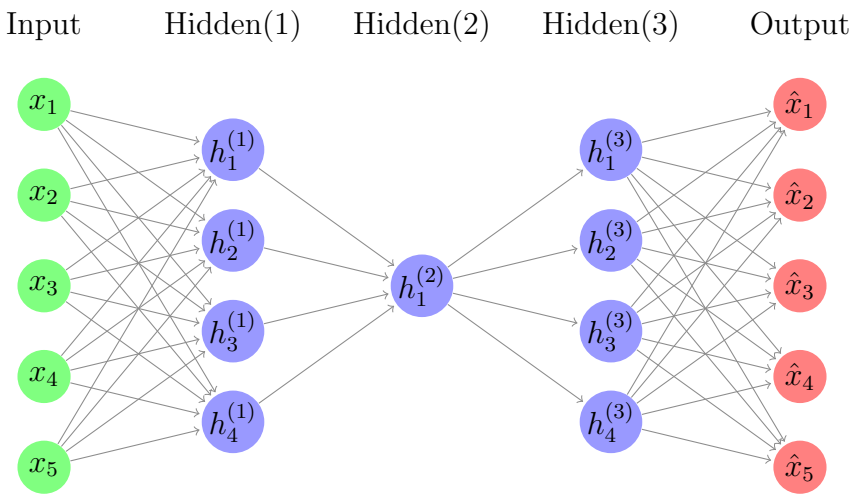


Figure 9: Graphical representation of an autoenconder

reasonable to hypothesize that inflation series *across* countries also exhibit nonlinear links. To allow for this possibility, we incorporate a third method, the autoencoder.

Autoencoders are the primary model for dimension reduction in the field of machine learning. They can be thought of as nonlinear neural network equivalents to PCA.[20] An autoencoder is a special type of unsupervised neural network in which the outputs attempt to approximate the input variables. The input variables pass through neurons in the hidden layer(s), creating a compressed representation of the input variables. Then, this compressed input is decoded (decompressed) into the output layer. The layer of interest, in our case, is the hidden layer with the smallest number of neurons since the neurons in this layer represent the latent nonlinear factors that we aim to extract. To illustrate the basic structure of an autoencoder, the diagram in Figure 9 portrays an autoencoder consisting of five inputs and three hidden layers with four, one and four neurons, respectively. The second hidden layer in this diagram represents the single latent factor we wish to extract. The layer preceding it is the encoding layer, while the layer that follows it is the decoding layer.

As with other neural networks, autoencoders can be written using the following recursive formula. Let $x_1, ..., x_{P_0}$, denote the P_0 dimensional set of input variables. One or more

[20]There is extensive literature recognizing the close connection between PCA and autoencoders, e.g., Baldi and Hornik (1989).

hidden layers follow. Each hidden layer l contains P_l neurons, each of which takes signals from the neurons in the previous hidden layer to generate a subsequent signal:

$$h_m^{(l)} = g\left(w_{m,0}^{(l)} + \sum_{j=1}^{P_{l-1}} w_{m,j}^{(l)} h_j^{(l-1)}\right) \quad \text{for } m = 1, \ldots, P_l;\ l = 1, \ldots, L, \tag{C.1}$$

where $h_m^{(l)}$ is the signal corresponding to the mth neuron in the lth hidden layer;[21] $w_{m,0}^{(l)}, w_{m,1}^{(l)}, \ldots, w_{m,P_{l-1}}^{(l)}$ are weights; and $g(\cdot)$ is a (nonlinear) activation function. The myriad of interactions among the inputs and the neurons in the network allows for complex nonlinear relationships in the autoencoder factor.

We use the leaky rectified linear unit (LReLU), $g(y) = \max(y, ay)$, as our nonlinear activation function where a is fixed at 0.3 (the `keras` default).[22] The final output of the autoencoder is:

$$h_m^{(L)} = w_{m,0}^{(L)} + \sum_{j=1}^{P_{L-1}} w_{m,j}^{(L)} h_j^{(L-1)}, \tag{C.2}$$

which shares the same dimension as the input.

For our application, we use an autoencoder with three hidden layers, where the second hidden layer only contains a single neuron (the nonlinear factor). We set the number of neurons of the first and third hidden layers to the square root of the size of the input layer, $\sqrt{P_0}$, following the rule proposed by Masters (1993).

Fitting (or training) an autoencoder requires estimating the weights. We train the autoencoder by minimizing the mean squared error between the input variables, $x_1, ..., x_{P_0}$, and the corresponding, reconstructed output variables, $\widehat{x}_1, ..., \widehat{x}_{P_0}$. We augment the objective function with an ℓ_1 penalty term equal to 0.001 to help guard against overfitting. We

[21] For the first hidden layer, $h_j^{(0)} = x_j$ for $j = 1, \ldots, P_0$.

[22] The LReLU activation function is a modification to the ReLU activation function, which is currently the most popular activation function in deep learning. LReLU was created to deal with the so-called "dying ReLU problem", where optimization in the network can fail as a result of many neurons taking a value of zero. Besides these advantages, recent papers in the deep learning literature (e.g. Xu et al. 2015) show that LReLU can result in performance gains compared to a ReLU activation function.

use the recently developed Adam SGD algorithm (Kingma and Ba 2015) to train the autoencoder, which we implement in `Python` using the `keras` package with a constant learning rate of 0.001. In the algorithm, we set the number of epochs to 700 and batch size to 32 (the `keras` default).

Irrespective of the method, the estimation of all factors is done recursively, using a rolling window that matches the estimation of Equation 3.1. The global factor uses all inflation series, while regional factors use only countries in each respective region as input.

C.3. Forecasting results with PCA and autoencoder factors

Table 10 shows the RMSE and MAD ratios of the RF model using factors computed as the cross-sectional mean vis-à-vis equivalent models that use PCA and AE to estimate the factors.

Table 10: RF forecasting results: alternative methodologies for factor construction

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	RMSE				MAD			
Model	$h=1$	$h=3$	$h=6$	$h=12$	$h=1$	$h=3$	$h=6$	$h=12$
PCA factors	1.001	1.001	1.004	1.003	1.000	1.005	1.001	1.102
AE factors	1.002	0.999	1.002	1.003	1.000	0.994	1.006	0.999

The table reports out-of-sample RMSE and MAD ratios of the RF model with mean factors (cf. Table 2) vis-à-vis equivalent models that use PCA and an Autoencoder to estimate the factors. The out-of-sample evaluation window runs from March 2000 to December 2019. Columns (1), (2), (3), and (4) report the ratios of average RMSE computed across all countries for $h = 1, 3, 6, 12$ while Columns (5), (6), (7), and (8) report the corresponding MAD ratios computed across all countries for $h = 1, 3, 6, 12$.

We generally find that the differences in performance are negligible, but mean factors and AE factors tend to produce better results than PCA. It is also interesting to note that the nonlinear AE factors do not seem to bring significant improvements over the simple average, implying that a nonlinear model such as RF already captures the nonlinearities required for improved performance.

D. Additional results

This appendix contains supplementary material that complement the results presented in the paper. The following is a brief summary of each section:

- While the map presented in the paper shows forecasting results of the Ensemble model for $h = 1$, Appendix D.1 shows the corresponding maps for $h = 3, 6, 12$ for completeness.

- In Appendix D.2, we show maps of the underlying country-specific RMSE ratios of the Ensemble model vis-à-vis the RW model to complement the binary representation of the forecasting results shown in the other maps.

- Appendix D.3 presents results obtained by computing averages of ratios, in contrast to the main results in the paper which were computed as ratios of averages.

- In Appendix D.4, we present for RMSE ratios to supplement the box plot figure of raw RMSE presented in the paper.

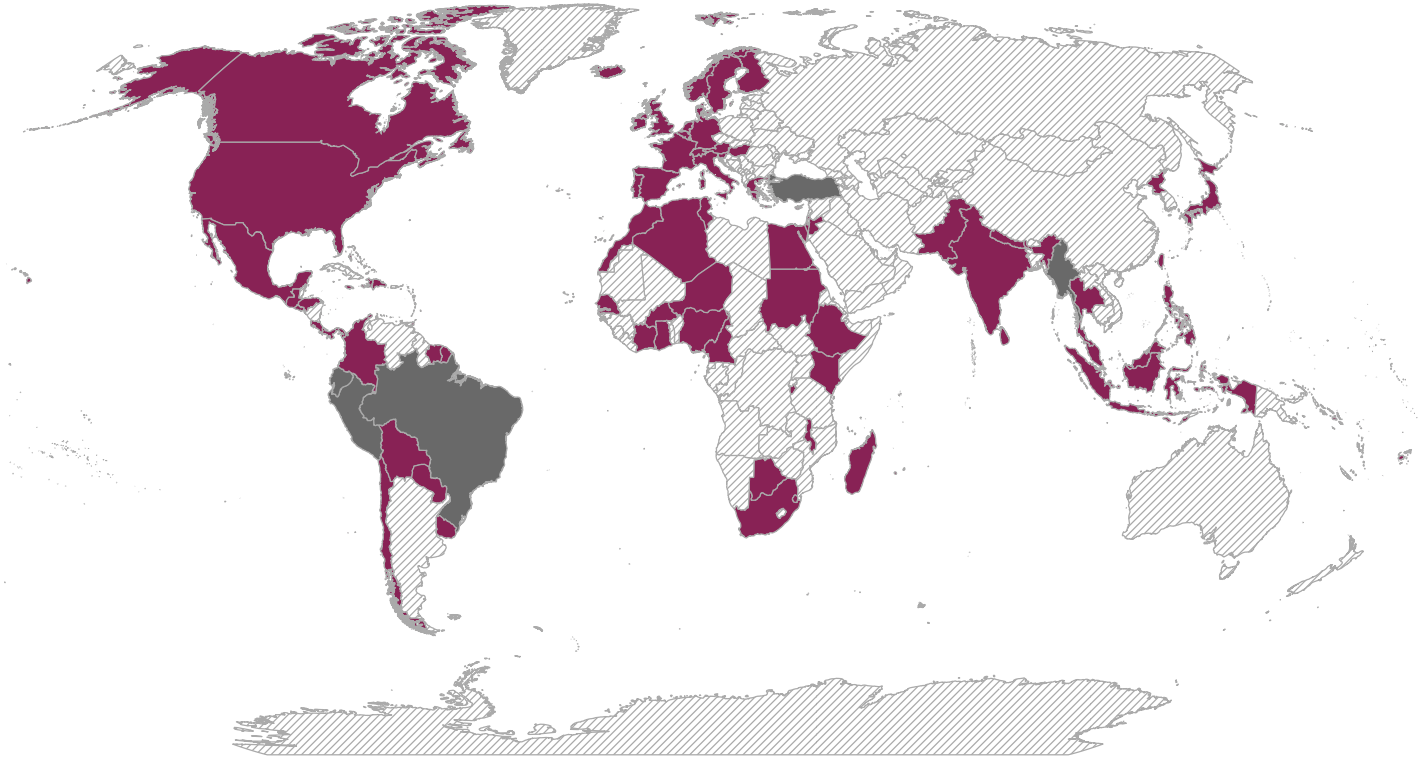
- Appendix D.5 shows cumulative sum of squared error difference (CSSED) plots for all models and specific countries at different points of the CSSED distribution.

- The CSSED figure presented in the paper is based on the Random Forest model. Appendix D.6 provides corresponding CSSED figures for the other models considered in the paper.

- Finally, Appendix D.7 presents variable importance (VI) results, where the aggregate importance of the nine predictor groups has been divided by the number of predictors in the groups to obtain the average importance of predictors in each group. This is in contrast to the VI results in the paper, where we show the aggregate importance of each predictor groups.

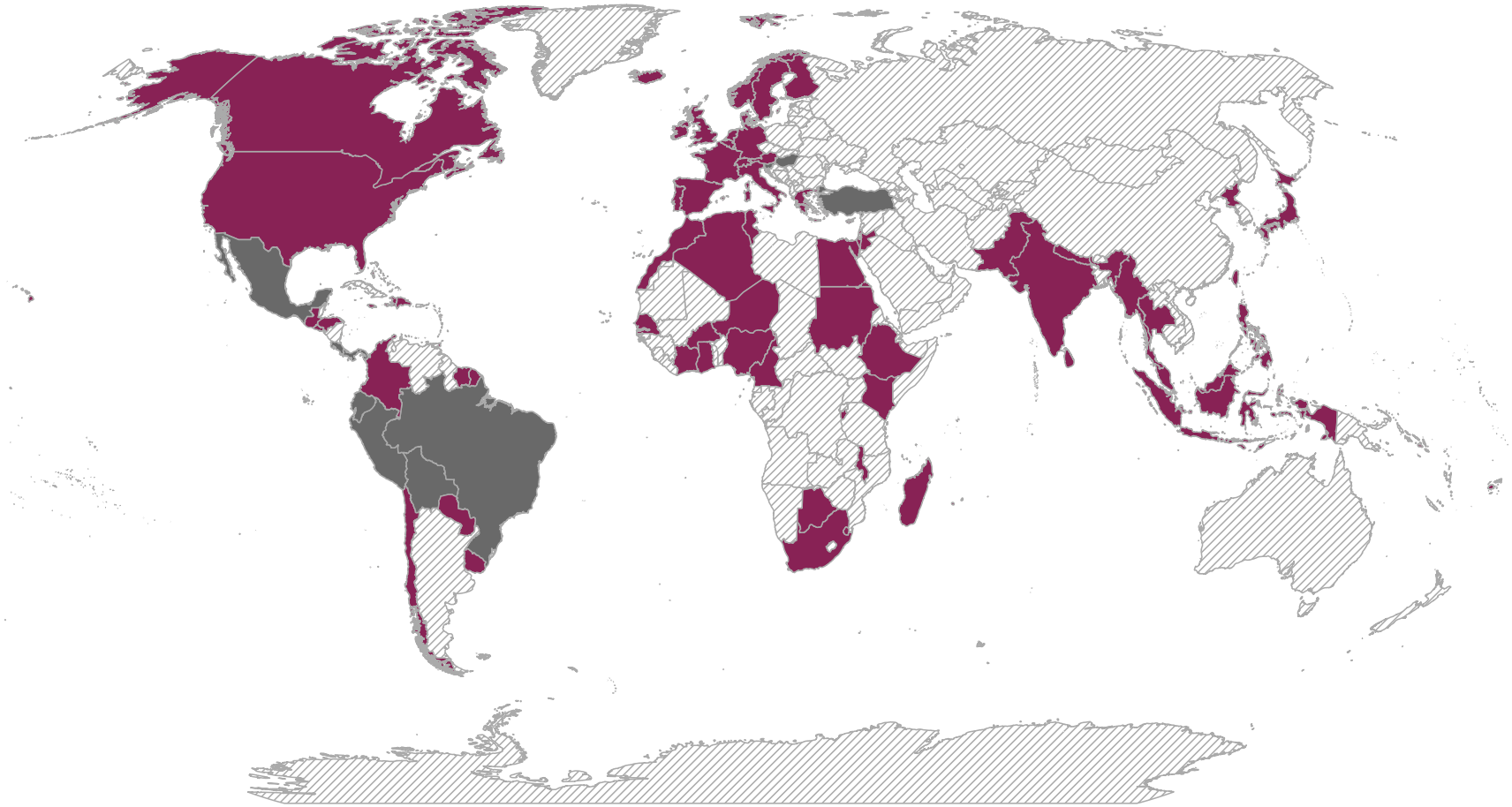
D.1. Forecast accuracy across countries - binary representation

Figure 10: Forecasting outperformance of an Ensemble of RF, NN, and EN models vis-à-vis the RW model for $h = 3$



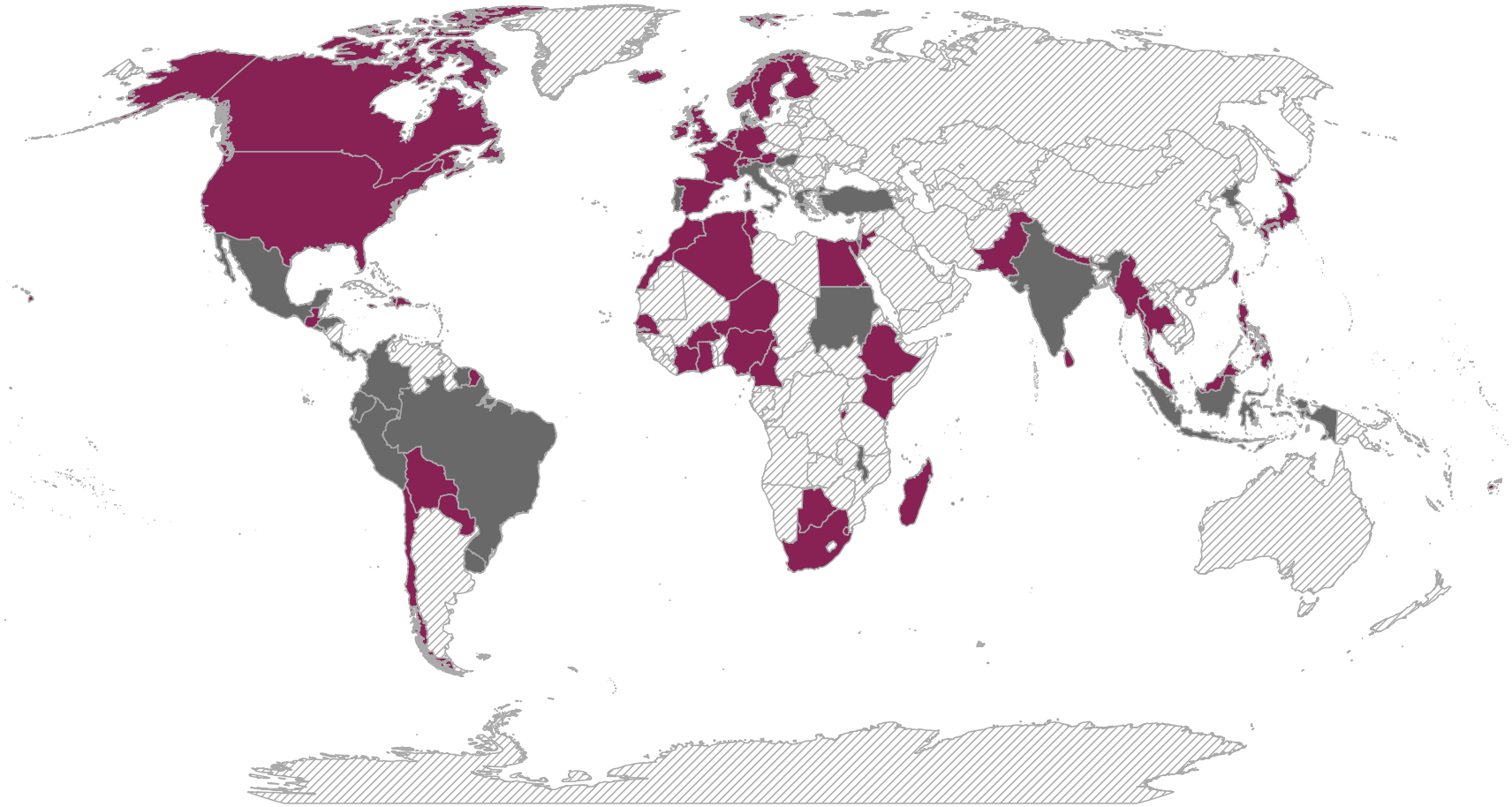
The map shows the outperformance of an equal-weighted Ensemble of RF, NN, and EN models vis-à-vis the RW model for $h = 3$. Countries colored in dark red have a RMSE ratio for the equal-weighted Ensemble vis-à-vis the RW model below 1, implying better forecasting performance for the three-month ahead forecast horizon. Countries colored in grey have a RMSE ratio above 1, implying worse forecasting performance. A total of 86 out of 91 countries have a ratio below 1.

Figure 11: Forecasting outperformance of an Ensemble of RF, NN, and EN models vis-à-vis the RW model for $h = 6$



The map shows the outperformance of an equal-weighted Ensemble of RF, NN, and EN models vis-à-vis the RW model for $h = 6$. Countries colored in dark red have a RMSE ratio for the equal-weighted Ensemble vis-à-vis the RW model below 1, implying better forecasting performance for the six-month ahead forecast horizon. Countries colored in grey have a RMSE ratio above 1, implying worse forecasting performance. A total of 81 out of 91 countries have a ratio below 1.

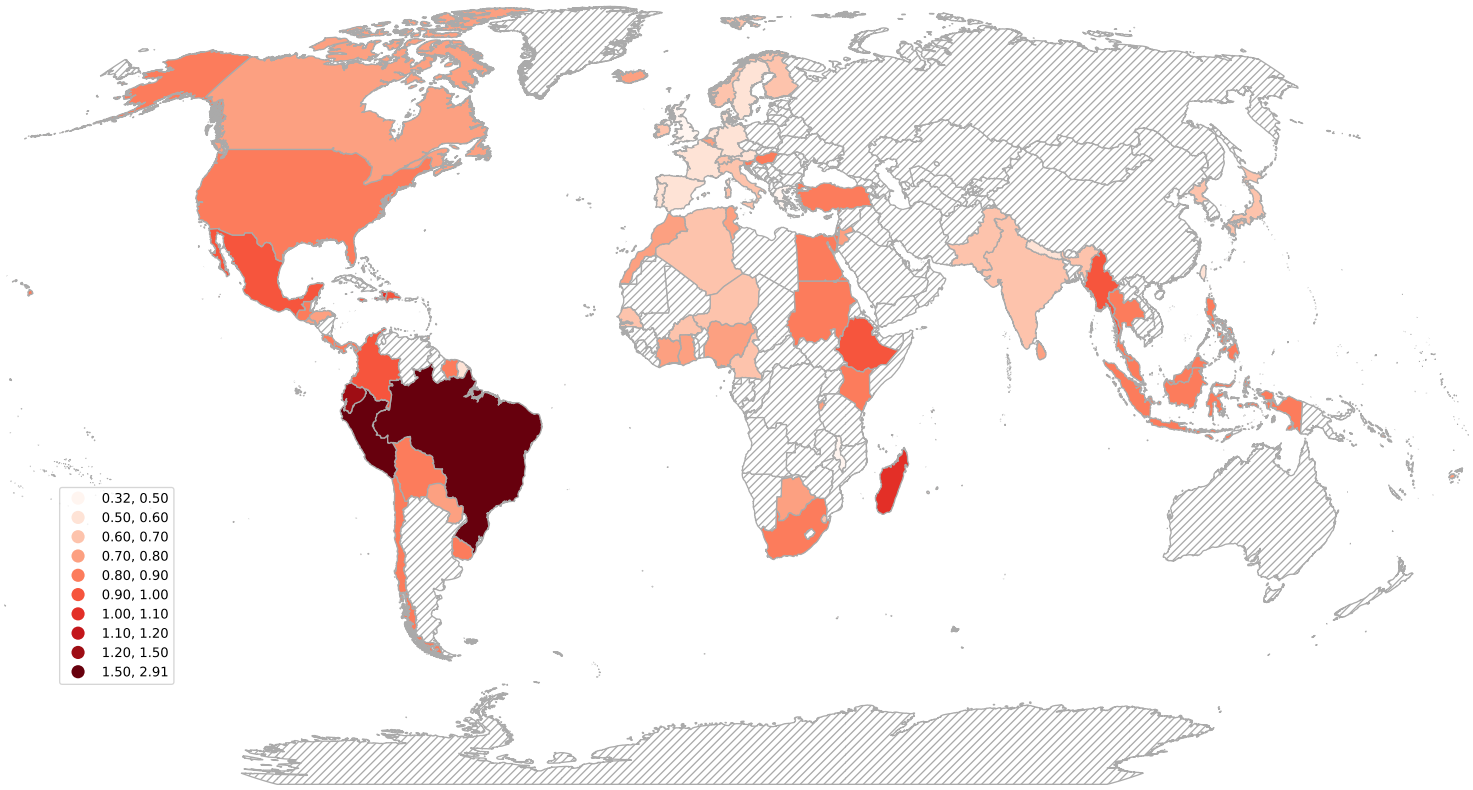
Figure 12: Forecasting outperformance of an Ensemble of RF, NN, and EN models vis-à-vis the RW model for $h = 12$



The map shows the outperformance of an equal-weighted Ensemble of RF, NN, and EN models vis-à-vis the RW model for $h = 12$. Countries colored in dark red have a RMSE ratio for the equal-weighted Ensemble vis-à-vis the RW model below 1, implying better forecasting performance for the one-year ahead forecast horizon. Countries colored in grey have a RMSE ratio above 1, implying worse forecasting performance. A total of 66 out of 91 countries have a ratio below 1.

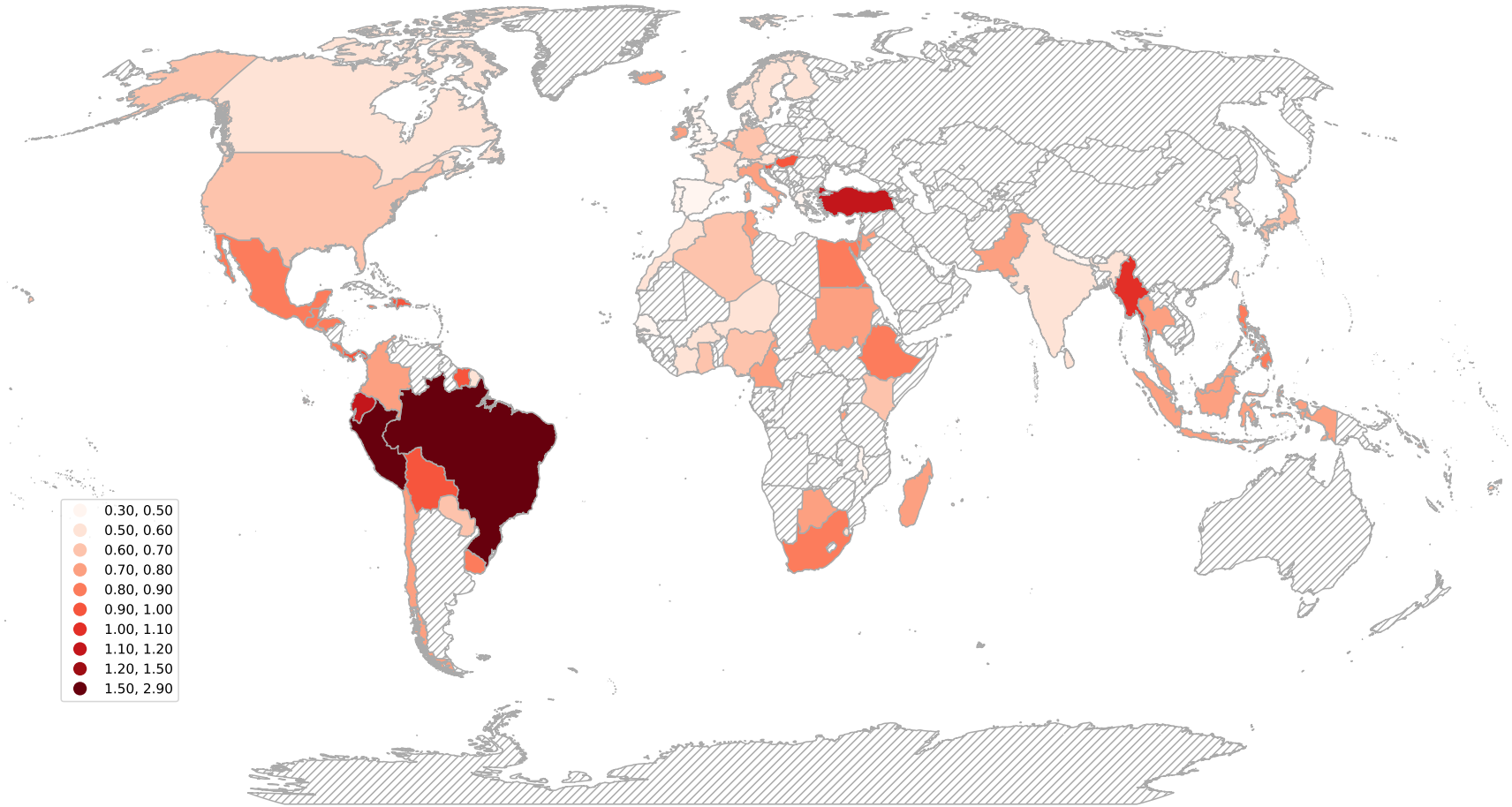
D.2. Forecast accuracy across countries

Figure 13: World map of country-specific RMSE ratios by the Ensemble model for $h = 1$



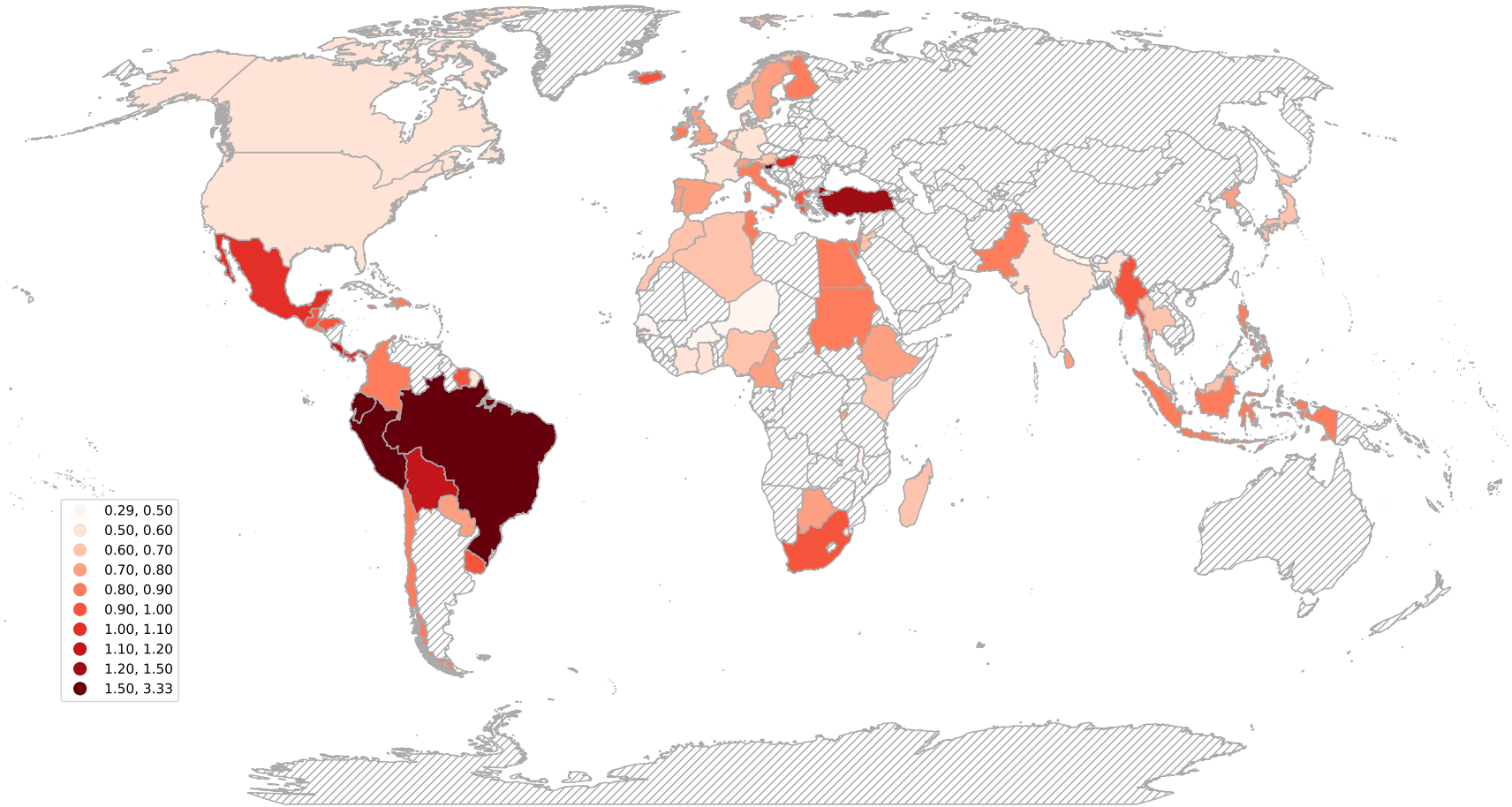
The map shows the country-specific RMSE ratios produced by the Ensemble model relative to the RW model for a forecast horizon of 1 month ahead. The countries are colored based on their respective RMSE ratios, where darker shades of red imply high RMSE ratios (higher forecast errors relative to the RW model), while lighter shades of red imply lower RMSE ratios (smaller forecast errors relative to the RW model).

Figure 14: World map of country-specific RMSE ratios by the Ensemble model for $h = 3$



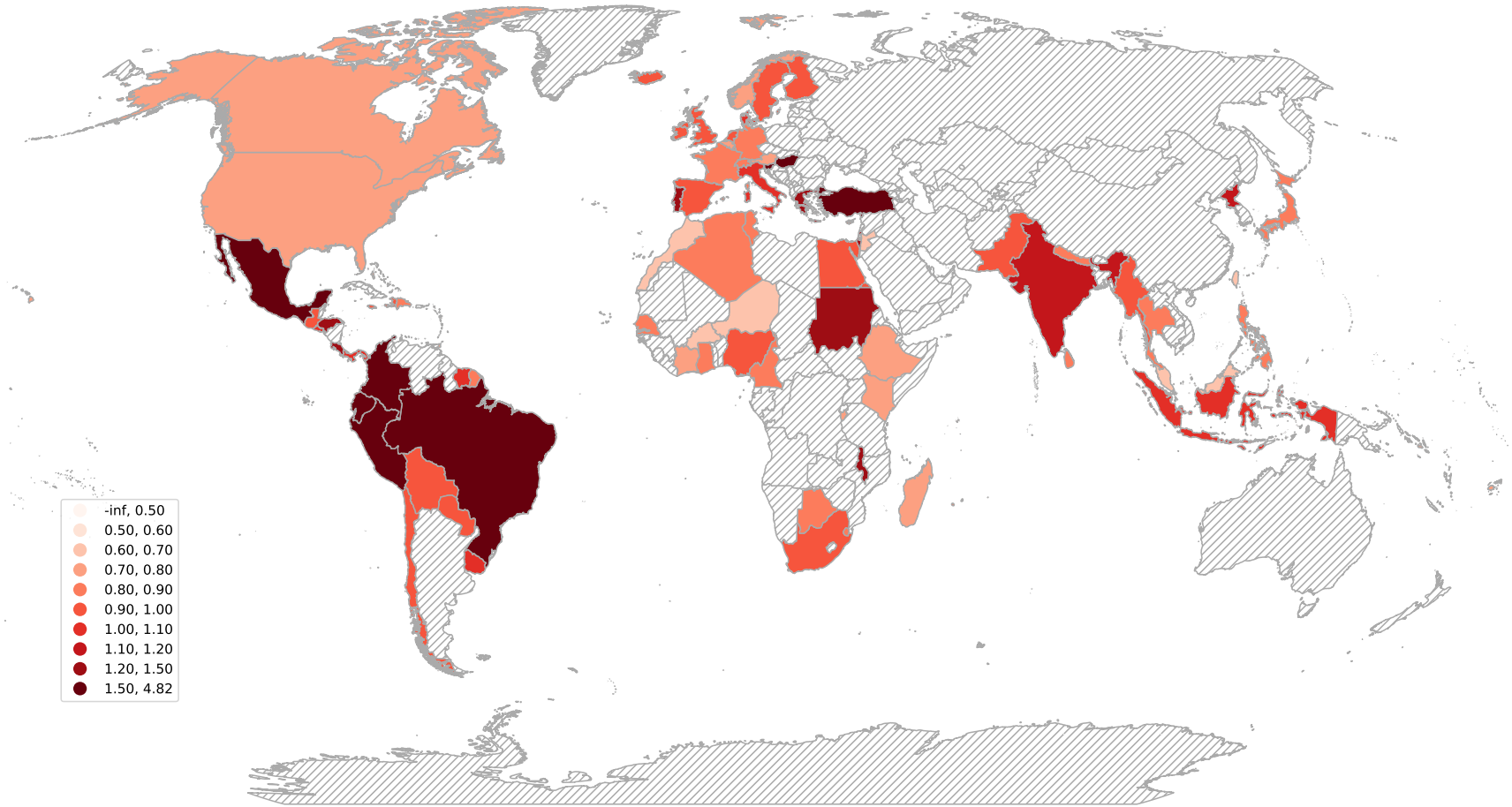
The map shows the country-specific RMSE ratios produced by the Ensemble model relative to the RW model for a forecast horizon of 3 months ahead. The countries are colored based on their respective RMSE ratios, where darker shades of red imply high RMSE ratios (higher forecast errors relative to the RW model), while lighter shades of red imply lower RMSE ratios (smaller forecast errors relative to the RW model).

Figure 15: World map of country-specific RMSE ratios by the Ensemble model for $h = 6$



The map shows the country-specific RMSE ratios produced by the Ensemble model relative to the RW model for a forecast horizon of 6 months ahead. The countries are colored based on their respective RMSE ratios, where darker shades of red imply high RMSE ratios (higher forecast errors relative to the RW model), while lighter shades of red imply lower RMSE ratios (smaller forecast errors relative to the RW model).

Figure 16: World map of country-specific RMSE ratios by the Ensemble model for $h = 12$



The map shows the country-specific RMSE ratios produced by the Ensemble model relative to the RW model for a forecast horizon of 12 months ahead. The countries are colored based on their respective RMSE ratios, where darker shades of red imply high RMSE ratios (higher forecast errors relative to the RW model), while lighter shades of red imply lower RMSE ratios (smaller forecast errors relative to the RW model).

D.3. Average of ratios

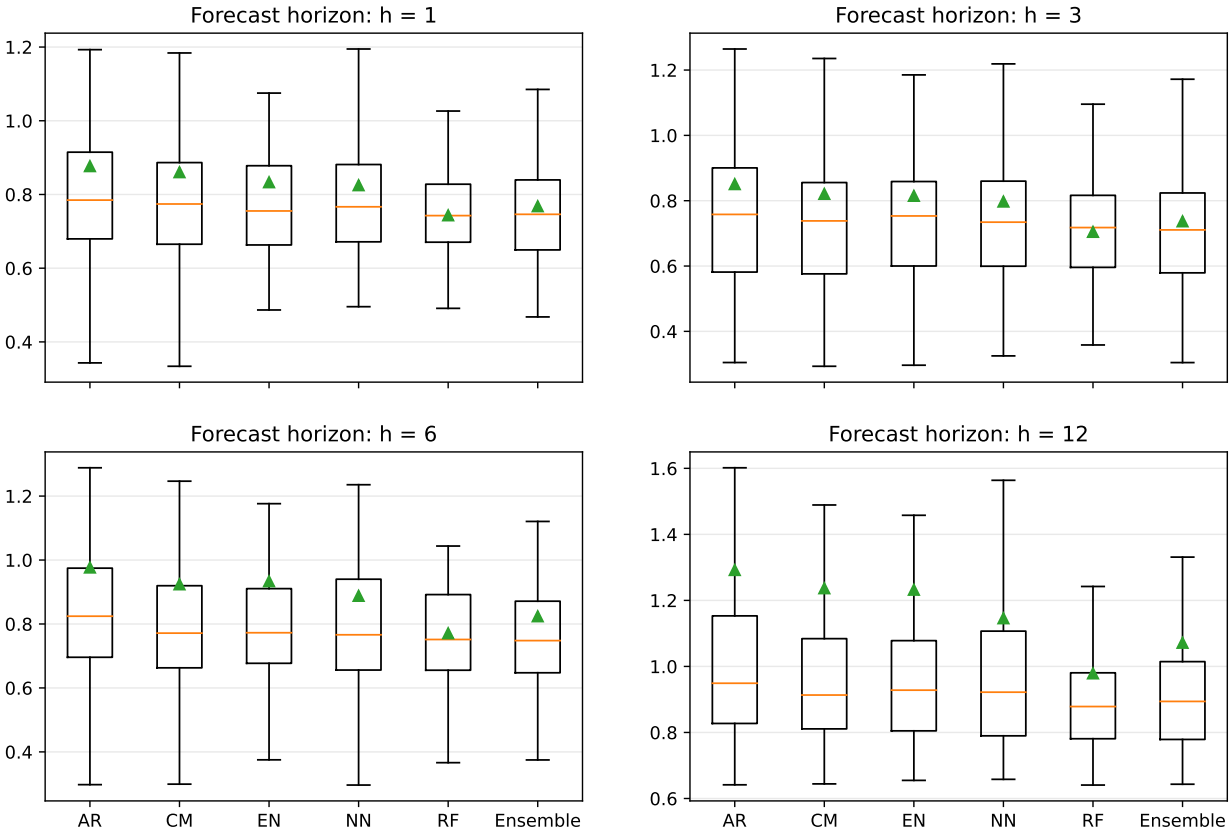
Table 11: Forecasting results: average of ratios

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
	RMSE				MAD				Frequency of outperformance		Multi-horizon SPA	
Model	$h=1$	$h=3$	$h=6$	$h=12$	$h=1$	$h=3$	$h=6$	$h=12$	% min. RMSE	% min. MAD	Mean	Share <0.05
RW	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	8%	10%		
AR	0.877	0.851	0.977	1.292	0.849	0.749	0.847	1.178	9%	15%	0.131	82%
CM	0.860	0.821	0.924	1.237	0.843	0.745	0.845	1.120	16%	11%	0.103	87%
EN	0.833	0.815	0.934	1.232	0.818	0.728	0.825	1.141	5%	4%	0.117	85%
NN	0.825	0.797	0.888	1.146	0.803	0.690	0.733	0.969	8%	13%	0.147	82%
RF	**0.744**	**0.705**	**0.771**	**0.979**	**0.733**	**0.649**	**0.711**	**0.950**	**28%**	22%	**0.042**	**95%**
Ensemble	0.768	0.737	0.824	1.072	0.756	0.657	0.720	0.967	26%	**24%**	0.071	90%

The table reports for each model several summary statistics computed across all 91 countries. Columns (1), (2), (3), and (4) report the average RMSE ratios across all countries for $h = 1, 3, 6, 12$ with the ratios computed between the model of interest and the RW model as a benchmark. Columns (5), (6), (7), and (8) report the corresponding average MAD ratios. Columns (9) and (10), respectively, report the number of times each model achieved the lowest RMSE and MAD across all country/horizon pairs as a share. Finally, columns (11) and (12) report the mean and median p-value across all countries of the average SPA test of Quaedvlieg (2021). The test is based on squared errors. The best results are highlighted in bold.

D.4. Results for RMSE ratios

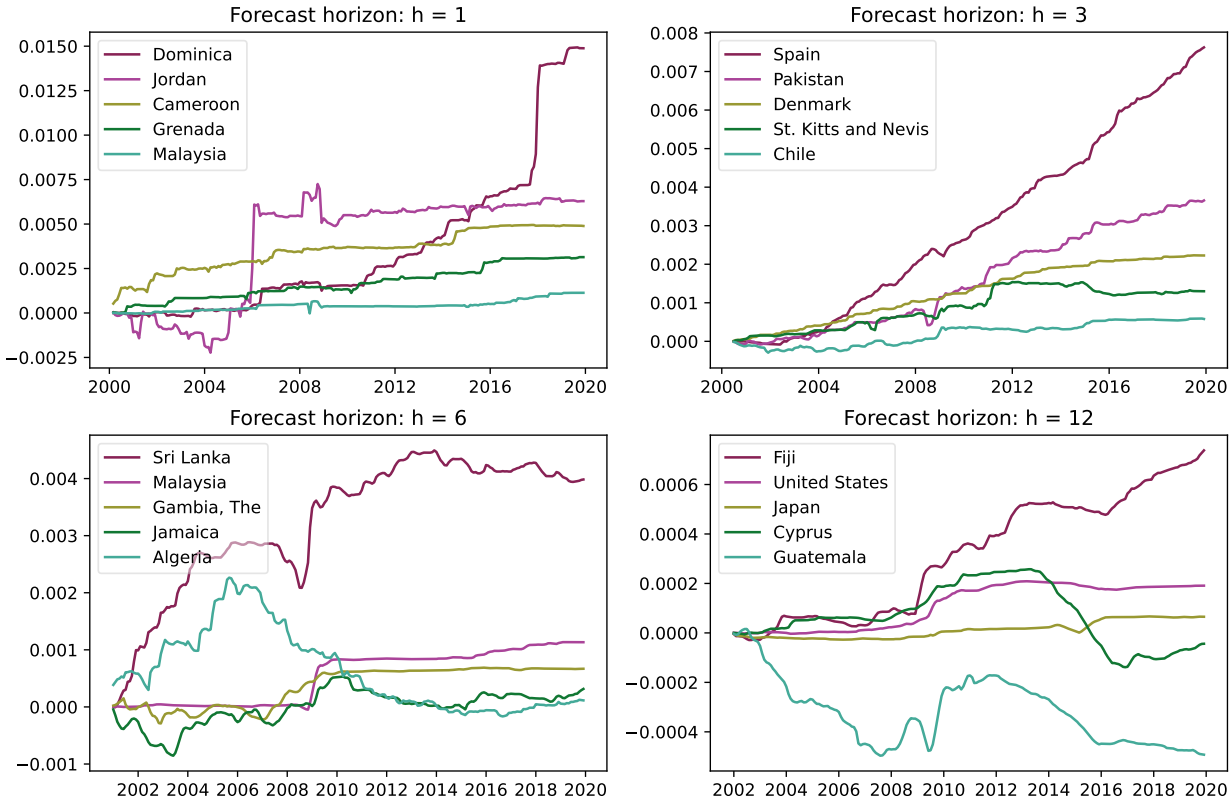
Figure 17: RMSE ratio box plots



The figure shows box plots for the RMSE ratios for all models, excluding the RW model, for $h = 1, 3, 6, 12$. The extremes of the box correspond to the 1st, Q1, and 3rd, Q3, quartile implying that 50% of the RMSE ratios are within these bounds. The upper whisker is computed as Q3 + 1.5 $*$ IQR, while the lower is computed as Q1 - 1.5 $*$ IQR, where IQR is the interquartile range. Any RMSE ratio falling outside the range of the whiskers is removed from the figure for readability. The green triangle reports the average RMSE ratio, also shown in Table 11 in Appendix D, while the orange horizontal line shows the median RMSE ratio.

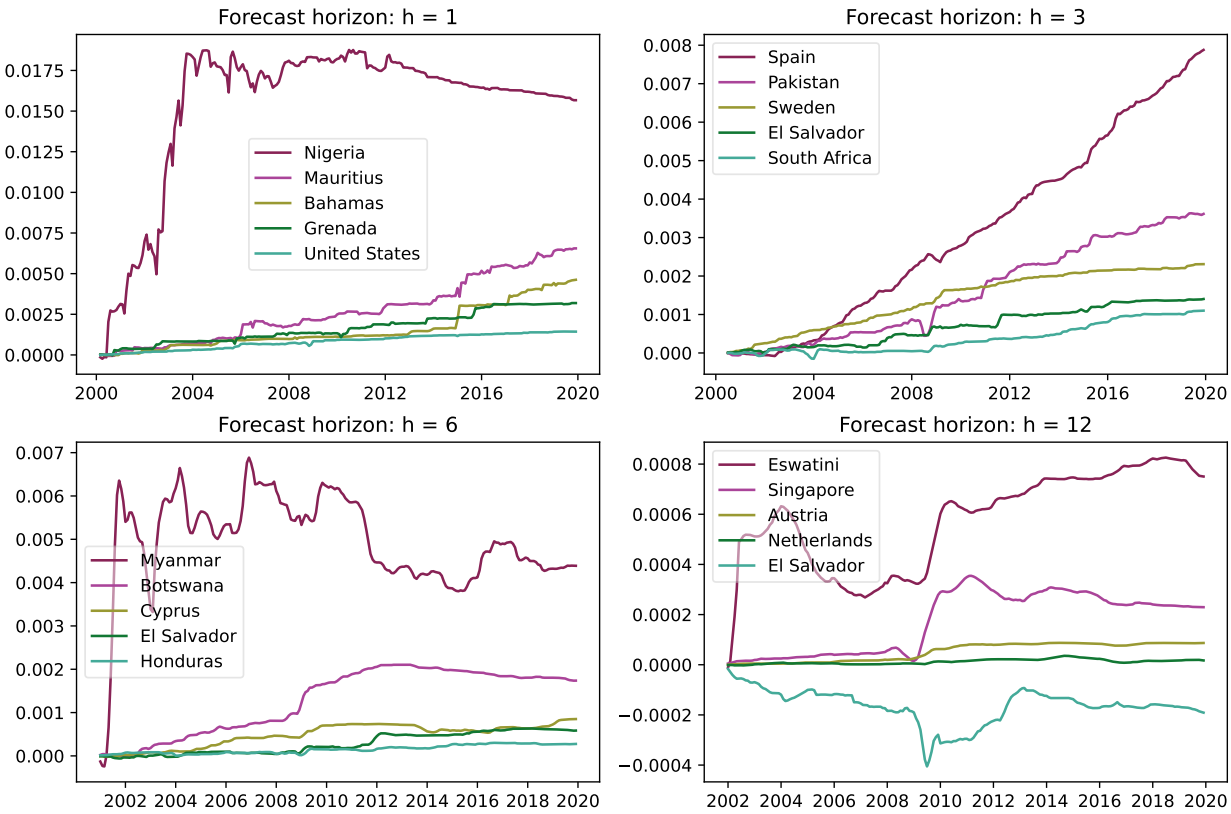
D.5. CSSED for specific countries

Figure 18: CSSED for the AR model for specific countries



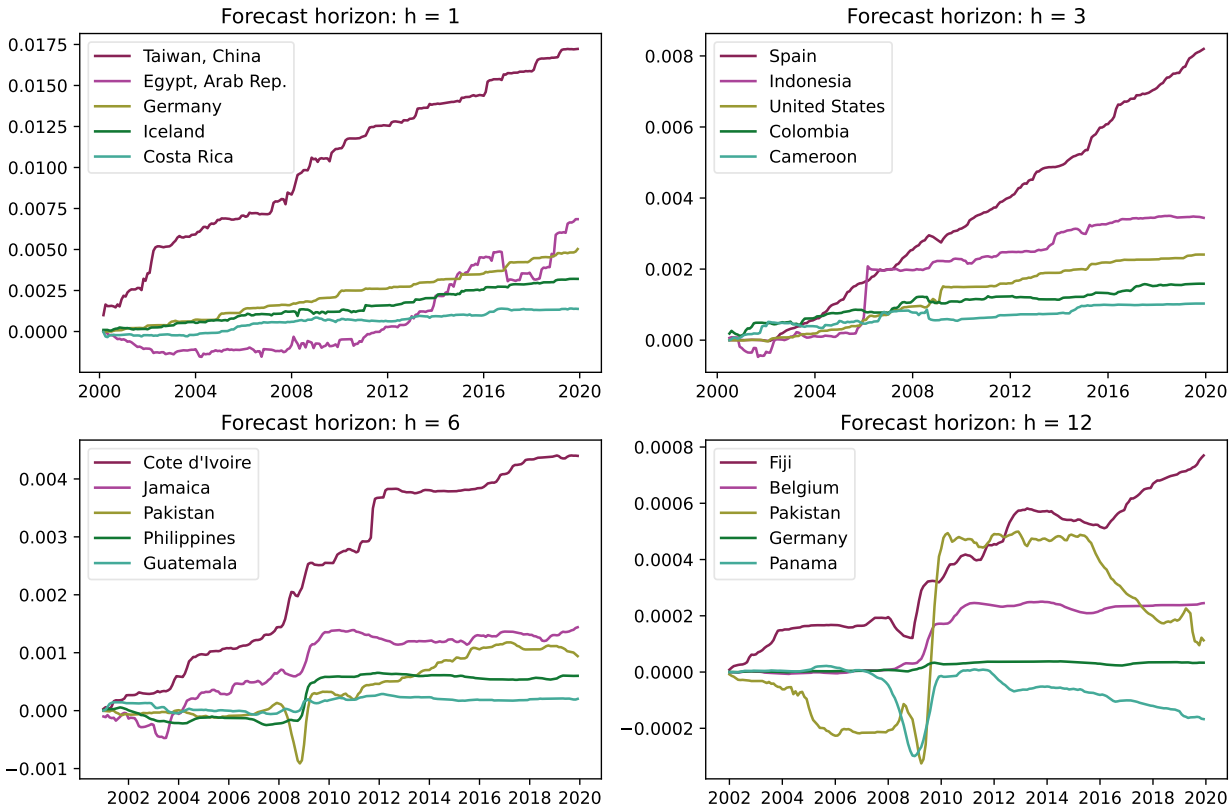
The figure shows the cumulative sum of squared differences (CSSED) for the AR model for $h = 1, 3, 6, 12$. We plot the country lying on the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles of the CSSED distribution across all countries for each horizon.

Figure 19: CSSED for the CM model for specific countries



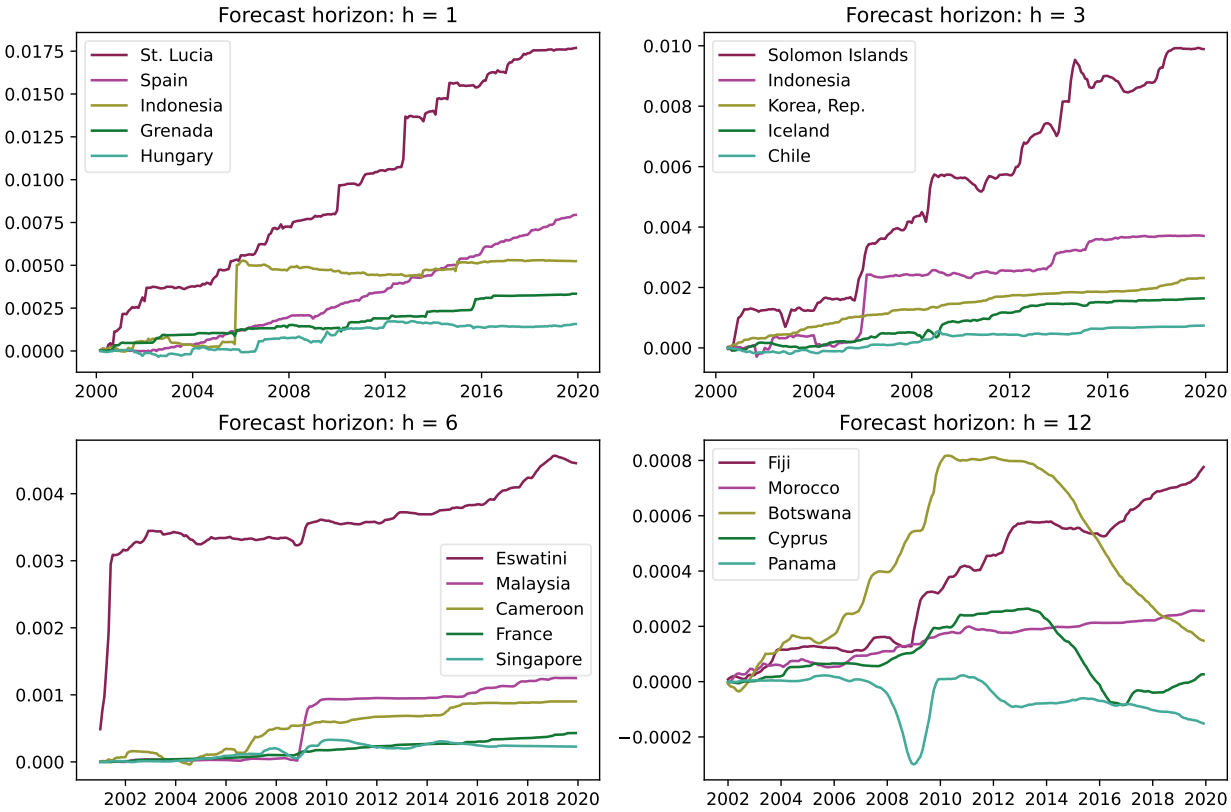
The figure shows the cumulative sum of squared differences (CSSED) for the CM model for $h = 1, 3, 6, 12$. We plot the country lying on the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles of the CSSED distribution across all countries for each horizon.

Figure 20: CSSED for the EN model for specific countries



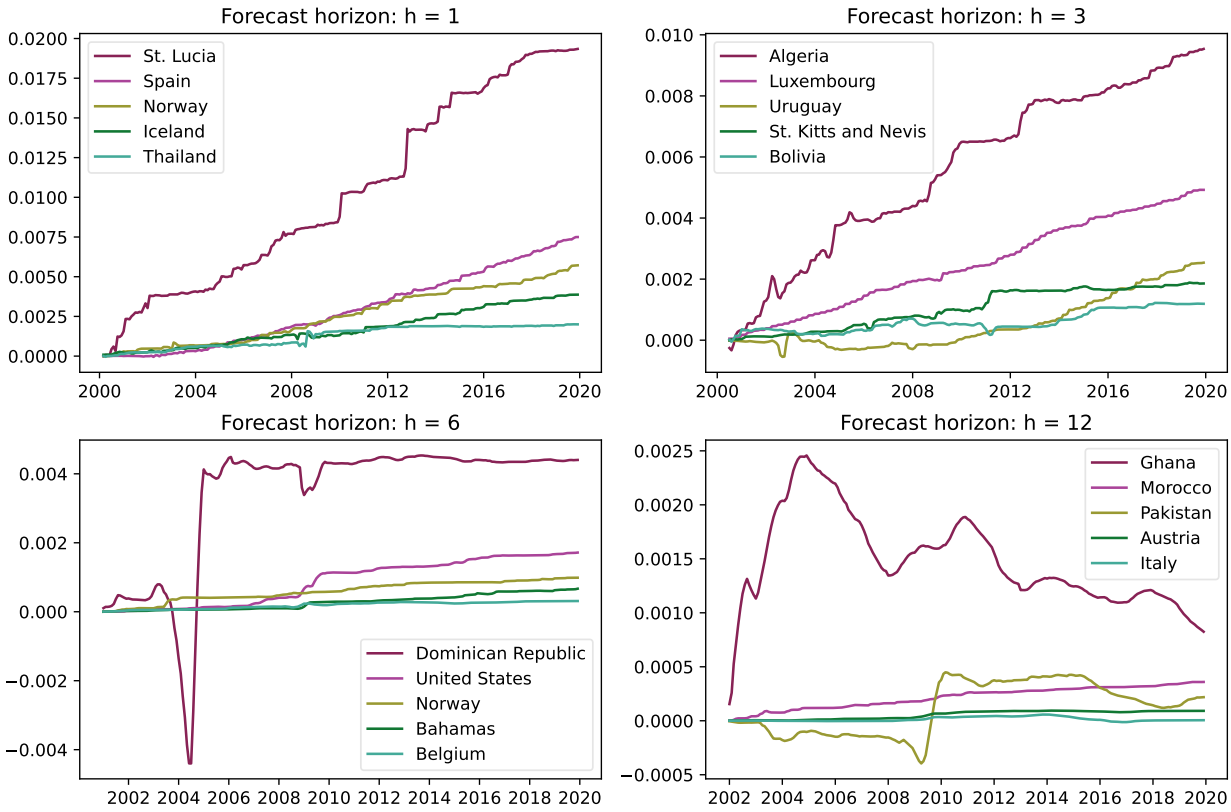
The figure shows the cumulative sum of squared differences (CSSED) for the EN model for $h = 1, 3, 6, 12$. We plot the country lying on the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles of the CSSED distribution across all countries for each horizon.

Figure 21: CSSED for the NN model for specific countries



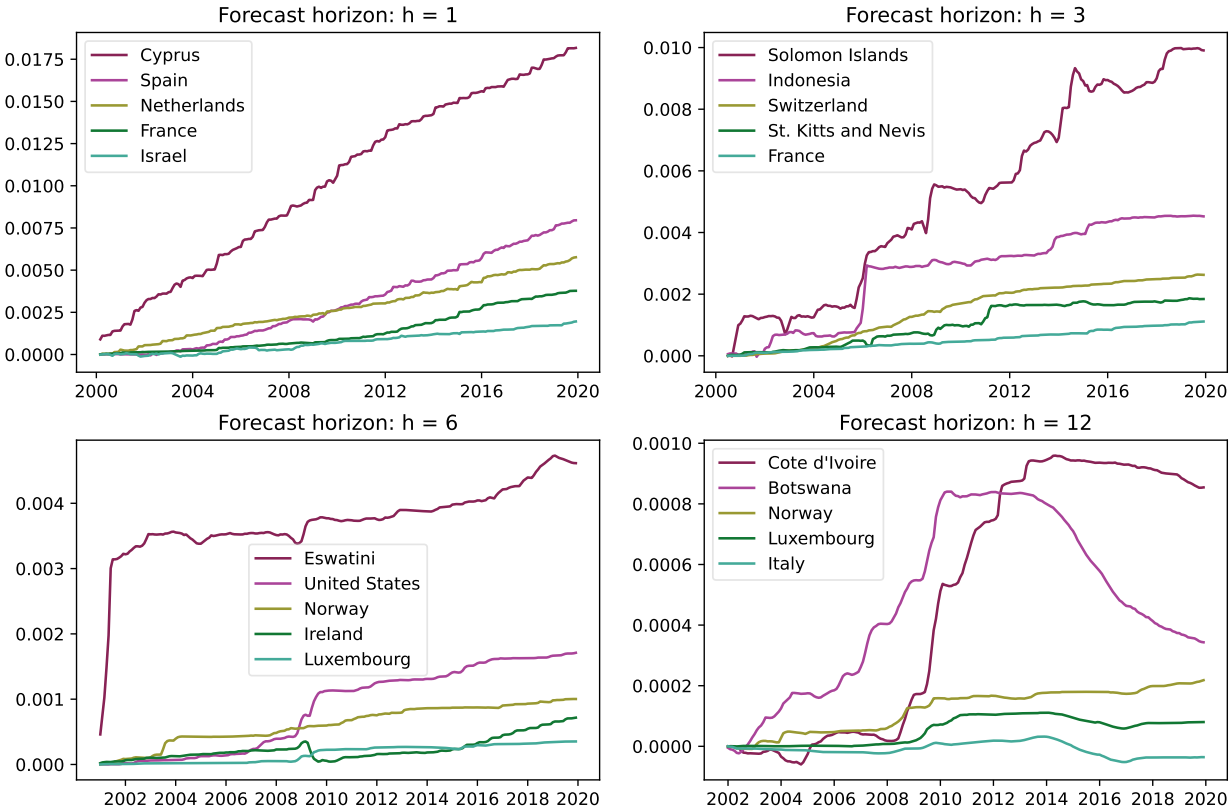
The figure shows the cumulative sum of squared differences (CSSED) for the NN model for $h = 1, 3, 6, 12$. We plot the country lying on the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles of the CSSED distribution across all countries for each horizon.

Figure 22: CSSED for the RF model for specific countries



The figure shows the cumulative sum of squared differences (CSSED) for the RF model for $h = 1, 3, 6, 12$. We plot the country lying on the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles of the CSSED distribution across all countries for each horizon.

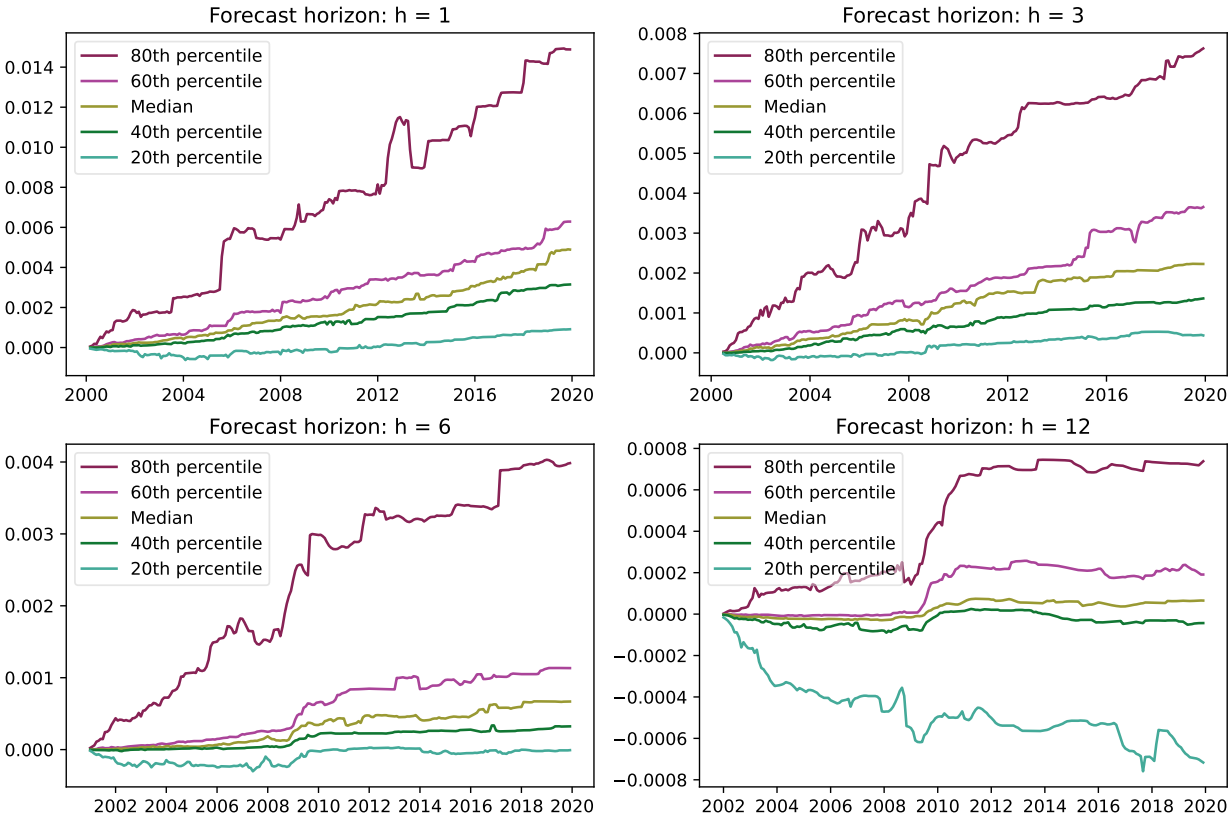
Figure 23: CSSED for the Ensemble model for specific countries



The figure shows the cumulative sum of squared differences (CSSED) for the Ensemble model for $h = 1, 3, 6, 12$. We plot the country lying on the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles of the CSSED distribution across all countries for each horizon.

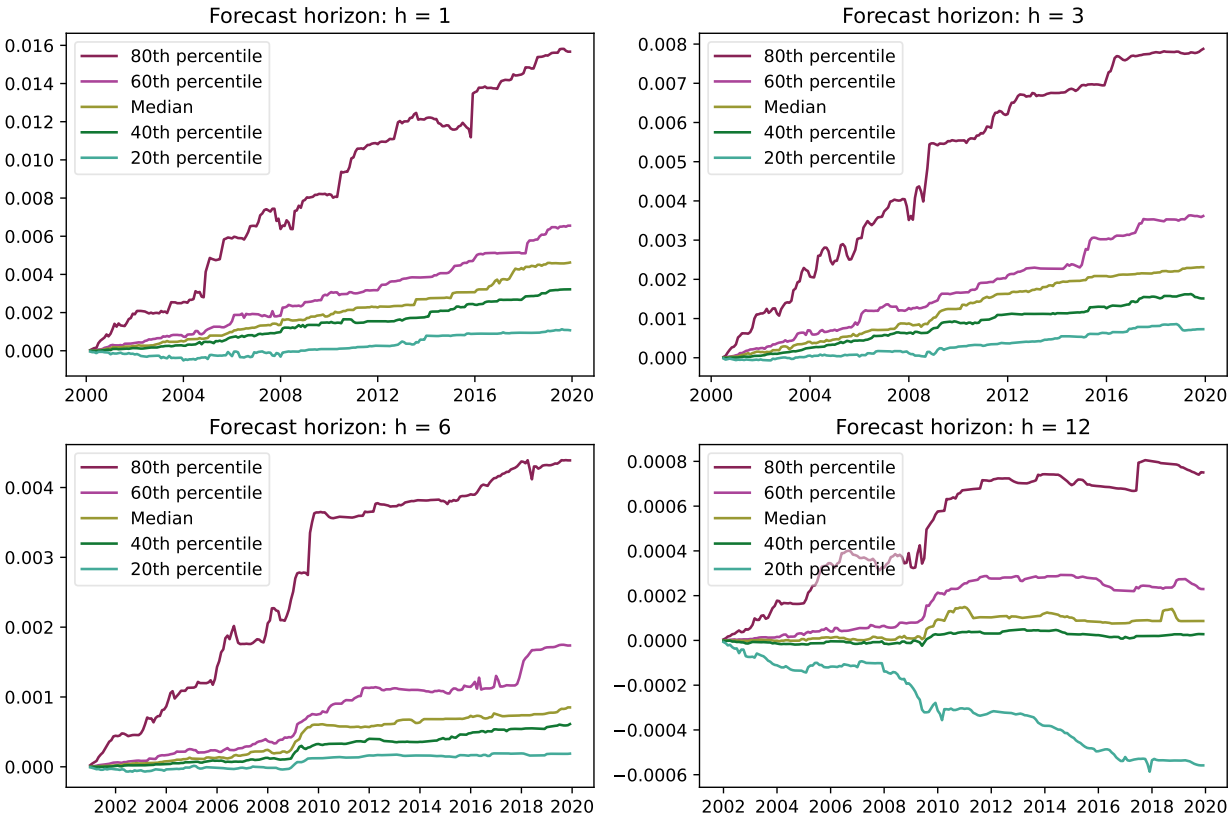
D.6. CSSED for the other models

Figure 24: CSSED for the AR model



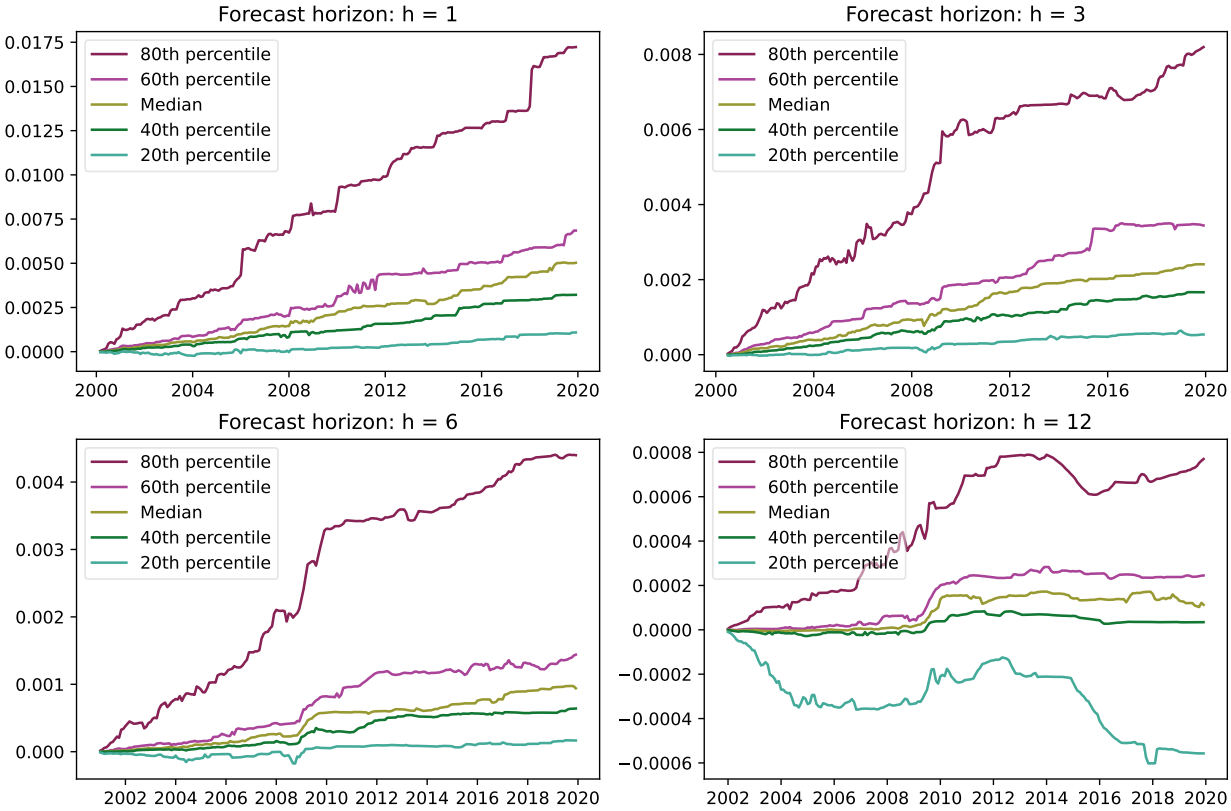
The figure shows the cumulative sum of squared differences (CSSED) for the AR model for $h = 1, 3, 6, 12$. We plot various percentiles of the CSSED distribution through time. A single line in the figure thus does not portray a specific country but instead a specific point in the CSSED distribution through time. We plot the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles for each horizon.

Figure 25: CSSED for the CM model



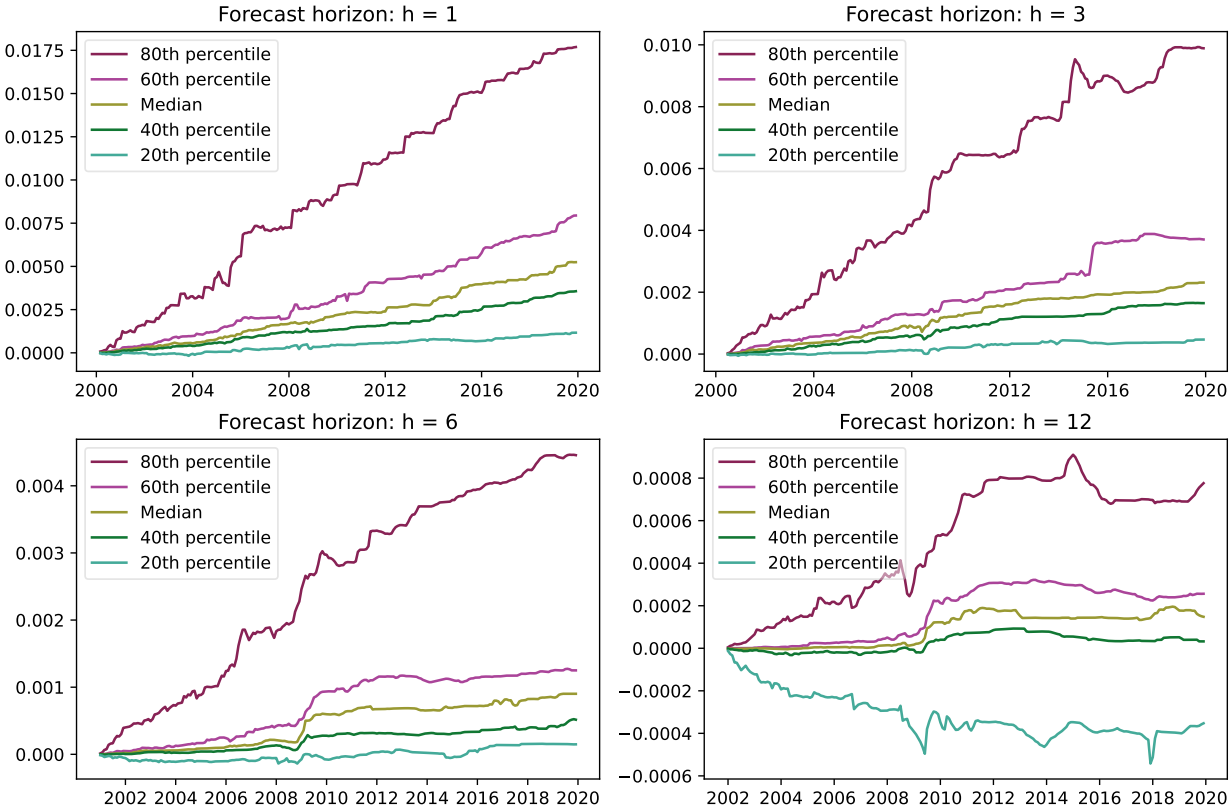
The figure shows the cumulative sum of squared differences (CSSED) for the CM model for $h = 1, 3, 6, 12$. We plot various percentiles of the CSSED distribution through time. A single line in the figure thus does not portray a specific country but instead a specific point in the CSSED distribution through time. We plot the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles for each horizon.

Figure 26: CSSED for the EN model



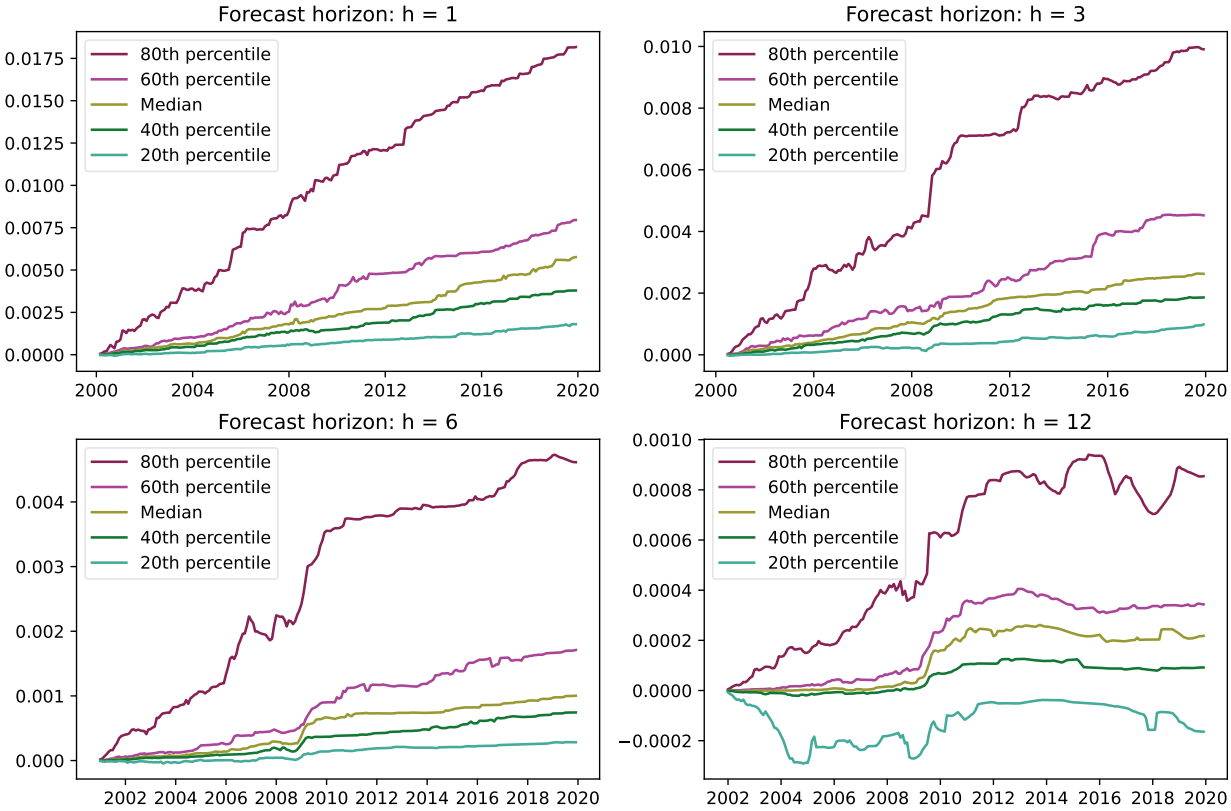
The figure shows the cumulative sum of squared differences (CSSED) for the EN model for $h = 1, 3, 6, 12$. We plot various percentiles of the CSSED distribution through time. A single line in the figure thus does not portray a specific country but instead a specific point in the CSSED distribution through time. We plot the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles for each horizon.

Figure 27: CSSED for the NN model



The figure shows the cumulative sum of squared differences (CSSED) for the NN model for $h = 1, 3, 6, 12$. We plot various percentiles of the CSSED distribution through time. A single line in the figure thus does not portray a specific country but instead a specific point in the CSSED distribution through time. We plot the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles for each horizon.

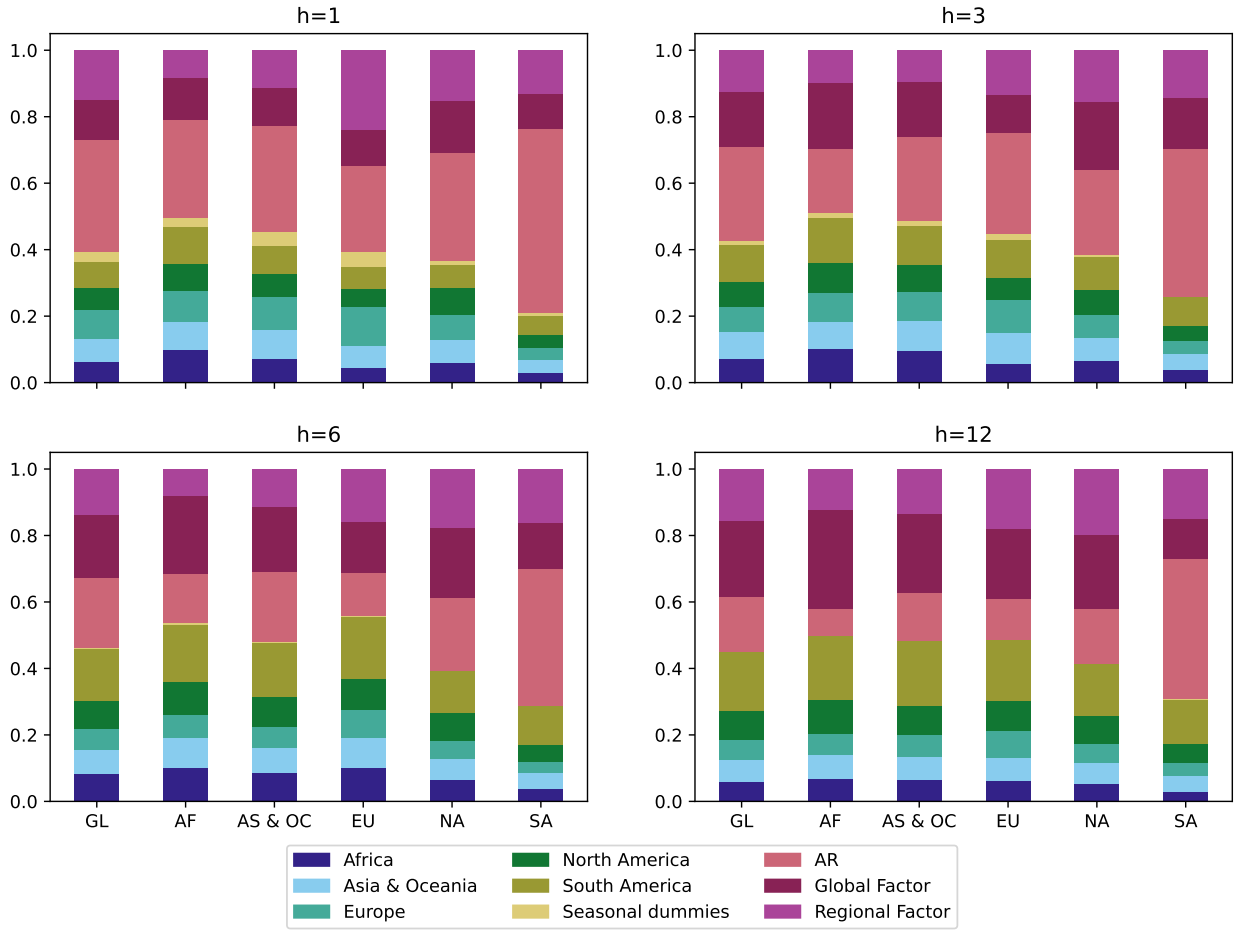
Figure 28: CSSED for the Ensemble model



The figure shows the cumulative sum of squared differences (CSSED) for the Ensemble model for $h = 1, 3, 6, 12$. We plot various percentiles of the CSSED distribution through time. A single line in the figure thus does not portray a specific country but instead a specific point in the CSSED distribution through time. We plot the 80th (dark red), 60th (purple), median (yellow), 40th (green) and 20th (turquoise) percentiles for each horizon.

D.7. Variable importance: average predictors in group

Figure 29: Variable importance: Average importance of predictors in group



The figure shows the average variable importance (VI) measure of the predictors in each group for the RF model at forecast horizons of $h = 1, 3, 6, 12$. The group to which the target variable belongs is shown on the x-axis. The groups are Global (GL), Africa (AF), Asia & Oceania (AS & OC), Europe (EU), North America (NA), and South America (SA). The VI measure is computed using Equation 3.9. The values are scaled to sum to one.

References for the appendix

Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks* 2:1, 53–58.

Binner, J. M., C. T. Elger, B. Nilsson, and J. A. Tepper (2006). Predictable non-linearities in US inflation. *Economics letters* 93:3, 323–328.

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

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

Coulombe, P., M. Leroux, D. Stevanovic, and S. Surprenant (Aug. 2020). How is Machine Learning Useful for Macroeconomic Forecasting?

Cuestas, J. C. and B. Harrison (2010). Inflation persistence and nonlinearities in Central and Eastern European countries. *Economics Letters* 106:2, 81–83.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer.

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

Hyndman, R. and G. Athanasopoulos (2018). *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts.

Kingma, D. P. and J. Ba (2015). *Adam: A Method for Stochastic Optimization*. Third Annual International Conference on Learning Representations. San Diego.

Masters, T. (1993). *Practical neural network recipes in C++*. Academic Press Professional, Inc.

Medeiros, M. C., G. F. Vasconcelos, Á. 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.

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

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58:1, 267–288. ISSN: 00359246.

Xu, B., N. Wang, T. Chen, and M. Li (2015). *Empirical Evaluation of Rectified Activations in Convolutional Network*.