International Journal of Forecasting 36 (2020) 86–92

Contents lists available at ScienceDirect

International Journal of Forecasting

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

FFORMA: Feature-based forecast model averaging

Pablo Montero-Manso *, George Athanasopoulos, Rob J. Hyndman, Thiyanga S. Talagala

Department of Econometrics and Business Statistics, Monash University, Australia

ARTICLE INFO

Keywords:
Time series features
Forecast combination
XGBoost
M4 competition
Meta-learning

ABSTRACT

We propose an automated method for obtaining weighted forecast combinations using time series features. The proposed approach involves two phases. First, we use a collection of time series to train a meta-model for assigning weights to various possible forecasting methods with the goal of minimizing the average forecasting loss obtained from a weighted forecast combination. The inputs to the meta-model are features that are extracted from each series. Then, in the second phase, we forecast new series using a weighted forecast combination, where the weights are obtained from our previously trained meta-model. Our method outperforms a simple forecast combination, as well as all of the most popular individual methods in the time series forecasting literature. The approach achieved second position in the M4 competition.

1. Introduction

There are essentially two general approaches to forecasting time series: (i) generating forecasts from a single model; and (ii) combining forecasts from many models (forecast model averaging). There is a vast body of literature on the latter, motivated by the seminal work of Bates and Granger (1969) and followed by a plethora of empirical applications showing that combination forecasts are often superior to their individual counterparts (see, Clemen, 1989; Timmermann, 2006, for example). Combining forecasts using a weighted average is considered a successful way of hedging against the risk of selecting a misspecified model. A major challenge is in selecting an appropriate set of weights, and many attempts to do this have ended up worse than simply using equal weights — something that has become known as the "forecast combination puzzle" (see for example Smith & Wallis, 2009). We address the problem of selecting the weights by using a meta-learning algorithm based on time series features.

There have been several previous attempts to use time series features combined with meta-learning for forecasting, for both model selection and combination (see for example Kang, Hyndman, & Smith-Miles, 2017; Kück, Crone, & Freitag, 2016; Lemke & Gabrys, 2010; Prudêncio & Ludermir, 2004). Recently, Talagala, Hyndman, and Athanasopoulos (2018) proposed the FFORMS (Feature-based FORecast Model Selection) framework, which uses time series features combined with meta-learning for forecast-model selection. That is, features are used to select a single forecasting model. The present paper builds on this framework by using meta-learning to select the weights for a weighted forecast combination. All candidate forecasting methods are applied, and the weights to be used for combining them are chosen based on the features of each time series. We call this framework FFORMA (Feature-based FORecast Model Averaging). FFORMA resulted in the second most accurate point forecasts and prediction intervals amongst all competitors in the M4 competition.

* Corresponding author.
E-mail addresses: p.montero.manso@udc.es (P. Montero-Manso), george.athanasopoulos@monash.edu (G. Athanasopoulos), rob.hyndman@monash.edu (R.J. Hyndman), thiyanga.talagala@monash.edu (T.S. Talagala).

https://doi.org/10.1016/j.ijforecast.2019.02.011
0169-2070/

The rest of the paper is organized as follows. Section 2 describes the FFORMA framework in a general sense. Section 3 then gives the details of our implementation of FFORMA in the M4 competition for generating both point and interval forecasts. This includes the required preprocessing steps, the set of features and forecast methods, and the specific implementation of the meta-learning model. We show empirical evidence on the performance of the approach in Section 4 by quantifying the difference between our proposed learning model and a traditional classifier approach. Section 4 also provides some final remarks and conclusions.

2. Methodology

2.1. Intuition and overview of FFORMA

The objective of our meta-learning approach is to derive a set of weights for combining the forecasts generated from a *pool of methods* (e.g., naïve, exponential smoothing, ARIMA, etc.). The FFORMA framework requires a set of time series that we refer to as the *reference set*. Each time series in the reference set is divided into a training period and a test period. A set of *time series features* are calculated from the training period (e.g., length of time series, strength of trend, autocorrelations, etc.), and these form the inputs to the meta-learning model. Each method in the pool is fitted to the training period, forecasts are generated over the test period, and *forecast errors* (the differences between actual and forecast values) are computed. Then, a summary forecast loss measure from a weighted combination forecast can be computed from these for any given set of weights.

The meta-learning model learns to produce weights for all methods in the pool, as a function of the features of the series to be forecast, by minimizing this summary forecast loss measure. Once the model has been trained, weights can be produced for a new series for which forecasts are required. It is assumed that the new series comes from a *generating process* that is similar to some of those that form the reference set.

A common meta-learning approach involves selecting the best method out of the pool of methods for each series; i.e., the method that produces the smallest forecast loss. This approach transforms the problem into a traditional classification problem by setting the individual forecasting methods as the classes and the best method as the target class for each time series. However, there may be other methods that produce similar forecast errors to the best method, so the specific class chosen is less important than the forecast error that results from each method. Furthermore, some time series are more difficult to forecast than others, and hence have more impact on the total forecast error. This information is lost if the problem is treated as classification.

Consequently, we do not train our meta-learning algorithm using a classification approach. Instead, we pose the problem as finding a function that assigns *weights* to each forecasting method, with the objective of minimizing the expected loss that would have been produced if the methods had been picked at random using these weights as probabilities. These are the weights in our weighted forecast combination. This approach is more general than classification, and can be thought of as classification with *per class weights* that vary per instance, combined with *per instance weights* that assign more importance to some series.

A flowchart of the FFORMA forecasting process can be seen in Fig. 1.

2.2. Algorithmic description

The operation of the FFORMA framework comprises two phases: (1) the offline phase, in which we train a meta-learner; and (2) the online phase, in which we use the pre-trained meta-learner to identify forecast combination weights for a new series. Algorithm 1 presents the pseudo-code of the proposed framework.

3. Implementation and application to the M4 competition

3.1. Reference set

The M4 dataset includes 100,000 time series of yearly, quarterly, monthly, weekly, daily and hourly data. Initially, all 100,000 series form the reference set. Each series is split into a training period and a test period. The length of the test period for each time series was set equal to the forecast horizon set by the competition. Series with training periods that comprised fewer than two observations, or series that were constant over the training period, were eliminated from the reference set.

3.2. Time series features

Table 1 provides a brief description of the features used in this experiment, F in Algorithm 1. The functions for calculating these are implemented in the `tsfeatures` R package by Hyndman, Wang et al. (2018). Most of the features (or variations of them) have been used previously in a forecasting context by Hyndman, Wang, and Laptev (2015) and Talagala et al. (2018), and are described in more detail there. The ARCH.LM statistic was calculated based on the Lagrange Multiplier test of Engle (1982) for autoregressive conditional heteroscedasticity (ARCH). The heterogeneity features 39–42 are based on two computed time series: the original time series is pre-whitened using an AR model, resulting in z; then, a GARCH(1,1) model is fitted to z to obtain the residual series, r.

Features that correspond only to seasonal time series are set to zero for non-seasonal time series. For the sake of generality, we have not used any domain-specific features, such as macro, micro, finance, etc., even though this information was available in the M4 data set.

3.3. Pool of forecasting methods

We considered nine methods implemented in the `forecast` package in R (Hyndman, Athanasopoulos et al., 2018) for the pool of methods, P in Algorithm 1:

1. naïve (`naive`);
2. random walk with drift (`rwf` with drift=TRUE);

Algorithm 1 The FFORMA framework: Forecast combination based on meta-learning

OFFLINE PHASE: TRAIN THE LEARNING MODEL

Inputs:

$\{x_1, x_2, \ldots, x_N\}$: N observed time series that form the reference set.
F: a set of functions for calculating time series features.
M: a set of forecasting methods in the pool, e.g., naïve, ETS, ARIMA, etc.

Output:

FFORMA meta-learner: A function from the extracted features to a set of M weights, one for each forecasting method.

Prepare the meta-data

for $n = 1$ to N: **do**
  Split x_n into a training period and test period.
  Calculate the set of features $\boldsymbol{f}_n \in F$ over the training period.
  Fit each forecasting method $m \in M$ over the training period and generate forecasts over the test period.
  Calculate forecast losses L_{nm} over the test period.
end for

Train the meta-learner, w

Train a learning model based on the meta-data and errors, by minimizing:

$$\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} \sum_{m=1}^{M} w(\boldsymbol{f}_n)_m L_{nm}.$$

ONLINE PHASE: FORECAST A NEW TIME SERIES

Input:

FFORMA meta-learner from offline phase.

Output:

Forecast the new time series x_{new}.

for each x_{new}: **do**
  Calculate features $\boldsymbol{f}_{new}$ by applying F.
  Use the meta-learner to produce $\boldsymbol{w}(\boldsymbol{f}_{new})$, an M-vector of weights.
  Compute the individual forecasts of the M forecasting methods in the pool.
  Combine the individual forecasts using $\boldsymbol{w}$ to generate final forecasts.
end for

3. seasonal naïve (`snaive`).
4. theta method (`thetaf`);
5. automated ARIMA algorithm (`auto.arima`);
6. automated exponential smoothing algorithm (`ets`);
7. TBATS model (`tbats`);
8. STLM-AR seasonal and trend decomposition using loess with AR modeling of the seasonally adjusted series (`stlm` with model function `ar`);
9. neural network time series forecasts (`nnetar`).

The R functions are given in parentheses. In all cases, the default settings were used. If any function returned an error when fitting the series (e.g. a series is constant), the `snaive` forecast method was used instead.

3.4. Forecast loss measure

The forecasting loss, L in Algorithm 1, was adapted from the overall weighted average (OWA) error described in the M4 competitor's guide M4 Competitor's Guide (2018), which combines the mean absolute scaled error and the symmetric mean absolute percentage error. For each series and method, the mean absolute scaled error and the symmetric mean absolute percentage error were divided by the respective errors of the Naive 2 method *over all series in the dataset* (i.e., MASE by the average MASE of Naive 2), and then added.

3.5. Meta-learning model implementation

We used the gradient tree boosting model of `xgboost` as the underlying implementation of the learning model (Chen & Guestrin, 2016). This is a state-of-the-art model that is computationally efficient and has shown a good performance in other problems. The great advantage of its application here is that we are able to customize it with our specific objective function.

The basic `xgboost` algorithm produces numeric values from the features, one for each forecasting method in our pool. We applied the softmax transform to these values prior to computing the objective function. This was implemented as a *custom objective function* in the `xgboost` framework.

`xgboost` requires a gradient and hessian of the objective function to fit the model. The *correct* hessian is prone

Table 1
Features used in the FFORMA framework.

Feature		Description	Non-seasonal	Seasonal
1	T	length of time series	✓	✓
2	trend	strength of trend	✓	✓
3	seasonality	strength of seasonality	–	✓
4	linearity	linearity	✓	✓
5	curvature	curvature	✓	✓
6	spikiness	spikiness	✓	✓
7	e_acf1	first ACF value of remainder series	✓	✓
8	e_acf10	sum of squares of first 10 ACF values of remainder series	✓	✓
9	stability	stability	✓	✓
10	lumpiness	lumpiness	✓	✓
11	entropy	spectral entropy	✓	✓
12	hurst	Hurst exponent	✓	✓
13	nonlinearity	nonlinearity	✓	✓
13	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓
14	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓
15	hwalpha	ETS(A,A,A) $\hat{\alpha}$	–	✓
16	hwbeta	ETS(A,A,A) $\hat{\beta}$	–	✓
17	hwgamma	ETS(A,A,A) $\hat{\gamma}$	–	✓
18	ur_pp	test statistic based on Phillips–Perron test	✓	✓
19	ur_kpss	test statistic based on KPSS test	✓	✓
20	y_acf1	first ACF value of the original series	✓	✓
21	diff1y_acf1	first ACF value of the differenced series	✓	✓
22	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓
23	y_acf10	sum of squares of first 10 ACF values of original series	✓	✓
24	diff1y_acf10	sum of squares of first 10 ACF values of differenced series	✓	✓
25	diff2y_acf10	sum of squares of first 10 ACF values of twice-differenced series	✓	✓
26	seas_acf1	autocorrelation coefficient at first seasonal lag	–	✓
27	sediff_acf1	first ACF value of seasonally differenced series	–	✓
28	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓
29	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓
30	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓
31	seas_pacf	partial autocorrelation coefficient at first seasonal lag	✓	✓
32	crossing_point	number of times the time series crosses the median	✓	✓
33	flat_spots	number of flat spots, calculated by discretizing the series into 10 equal-sized intervals and counting the maximum run length within any single interval	✓	✓
34	nperiods	number of seasonal periods in the series	–	✓
35	seasonal_period	length of seasonal period	–	✓
36	peak	strength of peak	✓	✓
37	trough	strength of trough	✓	✓
38	ARCH.LM	ARCH LM statistic	✓	✓
39	arch_acf	sum of squares of the first 12 autocorrelations of z^2	✓	✓
40	garch_acf	sum of squares of the first 12 autocorrelations of r^2	✓	✓
41	arch_r2	R^2 value of an AR model applied to z^2	✓	✓
42	garch_r2	R^2 value of an AR model applied to r^2	✓	✓

to numerical problems that need to be addressed in order for the boosting to converge. This is a relatively common problem, and one simple fix is to use an upper bound of the hessian by clamping its small values to a larger one. We computed a different upper bound of the hessian by removing some terms from the correct hessian. Although the two alternatives converged, the latter worked faster, requiring fewer boosting steps to converge. This not only increased the computational efficiency, but also generalized better, due to the production of a less complex set of trees in the final solution.

The general parameters of the meta-learning in Algorithm 1 were set as follows.

- $p(\boldsymbol{f}_n)_m$ is the output of the `xgboost` algorithm that corresponds to forecasting method m, based on the features extracted from series x_n.
- $w(\boldsymbol{f}_n)_m = \frac{\exp(p(\boldsymbol{f}_n)_m)}{\sum_m \exp(p(\boldsymbol{f}_n)_m)}$ is the transformation to weights of the `xgboost` output by applying the softmax transform.
- L_{nm} is the contribution of method m for the series n to the OWA error measure.
- $\bar{L}_n = \sum_{m=1}^{M} w(\boldsymbol{f}_n)_m L_{nm}$ is the weighted average loss function.
- $G_{nm} = \frac{\partial \bar{L}_n}{\partial p(\boldsymbol{f}_n)_m} = w_{nm}(L_{nm} - \bar{L}_n)$ is the gradient of the loss function.
- The hessian H_{nm} was approximated by our upper bound $\hat{H}_{nm}$:

$$H_{nm} = \frac{\partial G_n}{\partial p(\boldsymbol{f}_n)_m} \approx \hat{H}_n = w_n(L_n(1 - w_n) - G_n)$$

The functions G and $\hat{H}$ were passed to `xgboost` in order to minimize the objective function $\bar{L}$.

The results of `xgboost` depend particularly on its hyperparameters, such as the learning rate, number of boosting steps, maximum complexity allowed for the trees, or sub-sampling sizes. We limited the hyperparameter search space based on some initial results and rules-of-thumb and explored it using Bayesian optimization

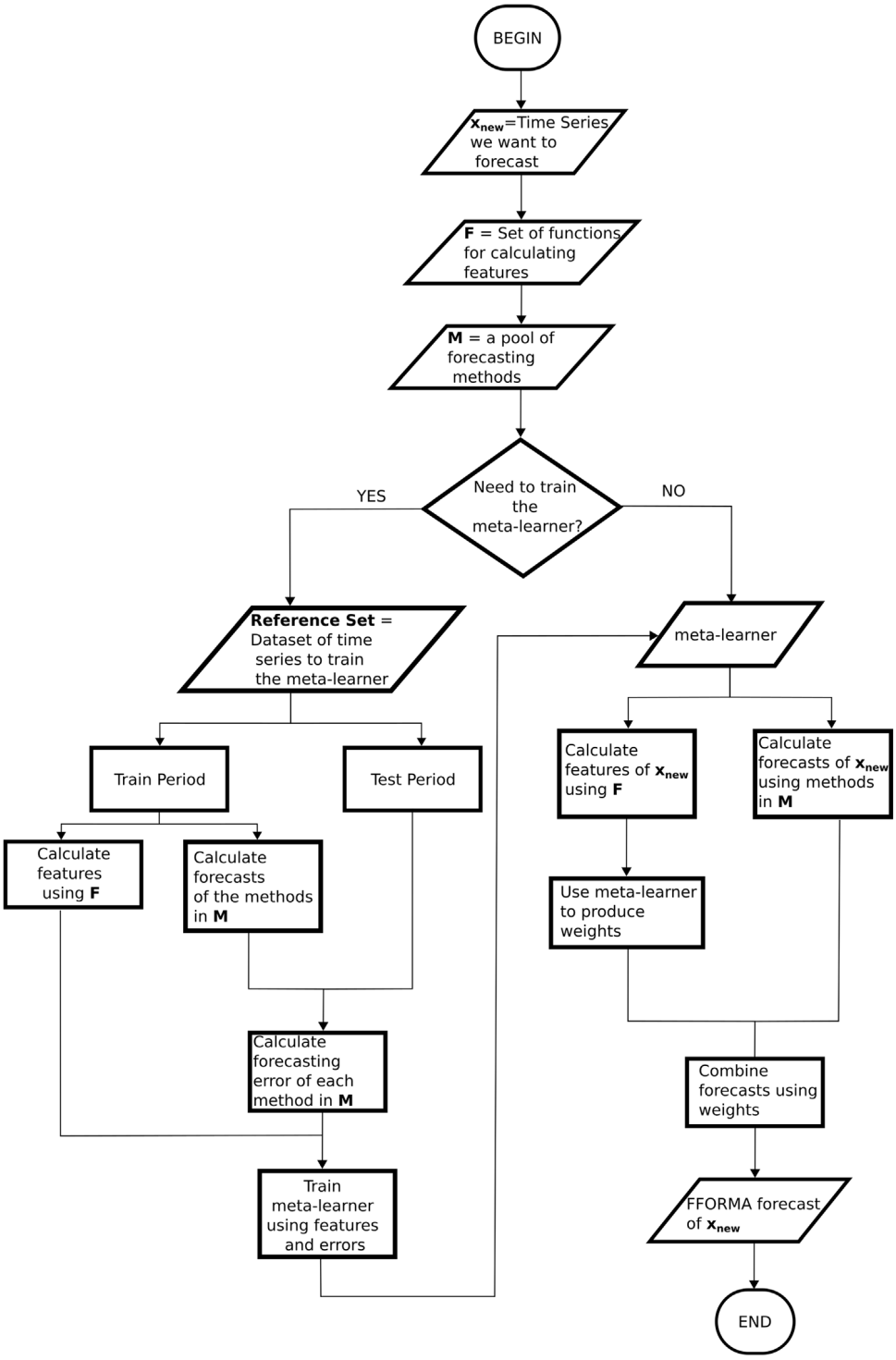

Fig. 1. Flowchart of the FFORMA forecasting process.

(implemented in the R package `rBayesianOptimization`, see Yan, 2016), measuring performance on a 10% holdout version of the reference set. We picked the simplest hyperparameter set from the top solutions to the exploration.

3.6. Prediction intervals

For each series x_{new}, we used the point forecast produced by our meta-learner as the center of the interval. Then, the 95% bounds of the interval were generated by

a linear combination of the bounds of three forecasting methods: naïve, theta and seasonal naïve. These methods were chosen from the initial pool purely in order to save computational time. The whole pool of methods should be used if computational time is not a constraint. The coefficients for the linear combination were calculated in a data-driven way over the M4 database. The complete procedure was as follows:

1. We divided the M4 dataset into two parts: A and B. We trained the FFORMA learner using the training periods of the series in part A and produced point forecasts for the test periods of the series in part B, and vice versa. This partitioning prevents the overfitting that can arise with extremely accurate point forecasts if the same dataset is used for both center and the interval bounds.
2. We computed the 95% *prediction interval radius* for the naïve, theta, and seasonal naïve methods. This is the difference between the 95% upper or lower bound and the point forecast for each forecast horizon, as we assume the intervals to be symmetric around the point forecast.
3. For each forecast horizon, we found the coefficients that minimized the MSIS of the interval, as defined in the M4 Competitor's guide (M4 Competitor's Guide, 2018), with the FFORMA point forecast as the center and a linear combination of the radii of naïve, theta, seasonal naïve forecasts as the interval. The minimization was done by gradient descent over the test period of the series.

This method produced a set of three coefficients for each prediction horizon in the M4 dataset, and these coefficients were the same independently of the series that we want to forecast. Unlike the point forecasts, these coefficients were not restricted to be probabilities.

4. Discussion and conclusions

We have presented an algorithm for forecasting using weighted averaging of a set of models. The objective function of the learning model assigns weights to forecasting methods in order to minimize the forecasting error that would be produced if we picked the methods at random using these weights as probabilities. This is in contrast to the way in which the final forecasts are produced, which is as a weighted average, not a selection.

However, these weights can be used as part of a model selection algorithm, if one picks the method that receives the largest weight. This can be useful for interpretability or computational reasons, at the cost of the forecasting performance.

We evaluated the impact of our contribution by comparing the average forecast error produced by FFORMA with that of a model selection approach. All implementation details were kept the same as in FFORMA; specifically, we used the same set of features, the same pool of forecasting methods, the same underlying implementation (`xgboost`) but with a standard cross-entropy loss, and the same hyperparameter search. This enabled us to measure the impact of the FFORMA loss function against that of a model selection approach, all other things being equal. We applied both FFORMA and the model selection approach to the M4 dataset and compared their overall point forecast errors. The average OWA error of the model selection approach was 10% larger than FFORMA. This improvement is due entirely to the use of the proposed model averaging rather than model selection. On the other hand, FFORMA deviates significantly from simple averaging: the latter produces a 14% increase in error for the same pool of methods. A simple cluster analysis of the weights shows that roughly 40% of the time series receive a weight profile that is similar to equal weights (a simple average) while the remaining 60% of the series have one of the methods in the pool clearly dominating.

FFORMA is also robust to changes in the pool of forecasting methods. The maximum increase in error when a single method from the original pool of nine methods is removed is 1%. This maximum occurs when removing the random walk with drift, which receives an average weight of 15% across all series in the M4. Also, the removal of any method results in an increased error compared to the original pool. Hence, there are no methods in the pool that have a negative impact on the error, even though some methods individually perform much worse than others when averaged over all the series. This suggests that the FFORMA algorithm is able to assign correct weights to the forecasting methods that are specialized for a specific type of series.

Thus, we believe that the good performance of the FFORMA can be attributed to three factors. First, the specific set of features and the learning model (`xgboost`) have been selected to give good forecasting results. Second, the weights allocated by FFORMA, which can range from simple averaging (equal weights) to a profile that assigns most of the importance to only a single method, allow FFORMA to outperform both simple averaging and method selection alternatives. Third, FFORMA adapts to the methods in the pool, making the specific pool of methods less critical than in other combination approaches.

One advantage of our approach is that its form is independent of the forecasting loss measure. Forecast errors enter the model as additional pre-calculated values. This allows FFORMA to adapt to arbitrary loss functions when models that minimize them directly would be restricted. For example, our approach can be applied to non-differentiable errors.

The source code for FFORMA is available at github.com/robjhyndman/M4metalearning.

Acknowledgments

George Athanasopoulos and Rob J Hyndman acknowledge support from the Australian Research Council grant DP1413220. Pablo Montero-Manso acknowledges support from Spanish Ministerio de Economía y Competitividad (grant MTM2014-52876-R), the Xunta de Galicia (Centro Singular de Investigación de Galicia accreditation ED431G/01 2016-2019 and Grupos de Referencia Competitiva ED431C2016-015) and the European Union (European Regional Development Fund - ERDF).

References

Bates, J. M., & Granger, C. W. J. (1969). The combination of forecasts. *The Journal of the Operational Research Society*, *20*(4), 451–468.

Chen, T., & Guestrin, C. (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.

Clemen, R. (1989). Combining forecasts: a review and annotated bibliography with discussion. *International Journal of Forecasting*, *5*, 559–608.

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

Hyndman, R. J., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., & Yasmeen, F. (2018) forecast: Forecasting functions for time series and linear models. R package version 8.3.

Hyndman, R. J., Wang, E., Kang, Y., & Talagala, T. (2018). tsfeatures: Time series feature extraction. R package version 0.1.

Hyndman, R. J., Wang, E., & Laptev, N. (2015). Large-scale unusual time series detection. In *2015 IEEE international conference on data mining workshop* (pp. 1616–1619). IEEE.

Kang, Y., Hyndman, R. J., & Smith-Miles, K. (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, *33*(2), 345–358.

Kück, M., Crone, S. F., & Freitag, M. (2016). Meta-learning with neural networks and landmarking for forecasting model selection — An empirical evaluation of different feature sets applied to industry data meta-learning with neural networks and landmarking for forecasting model selection. *International Joint Conference on Neural Networks*, 1499–1506.

Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. In *Subspace learning/selected papers from the european symposium on time series prediction Neurocomputing*, *73*(10), 2006–2016.

M4 Competitor's Guide (2018). https://www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-Competitors-Guide.pdf. (Accessed 26 September 2018).

Prudêncio, R., & Ludermir, T. (2004). Using machine learning techniques to combine forecasting methods. In *Australasian joint conference on artificial intelligence* (pp. 1122–1127). Springer.

Smith, J., & Wallis, K. F. (2009). A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics*, *71*(3), 331–355.

Talagala, T. S., Hyndman, R. J., & Athanasopoulos, G. (2018). Meta-learning how to forecast time series. Working Paper 6/18. Department of Econometrics & Business Statistics, Monash University.

Timmermann, A. (2006). Forecast combinations. In *Handbook of economic forecasting* (pp. 135–196). Amsterdam: North-Holland.

Yan, Y. (2016). rBayesianOptimization: Bayesian optimization of hyperparameters. R package version 1.1.0.

Pablo Montero-Manso is a Research Fellow in the Department of Econometrics and Business Statistics at Monash University. His research interests include Time Series Clustering and Classification, and Distributed Statistical Learning.

George Athanasopoulos is Professor in the Department of Econometrics and Business Statistics, Monash University, Australia. His research interests include forecasting hierarchical and grouped time series, multivariate time series analysis, and tourism economics. He is Associate Editor of the International Journal of Forecasting and on the Editorial Board of the Journal of Travel Research.

Rob J. Hyndman is Professor of Statistics at Monash University, Australia, and Editor-in-Chief of the International Journal of Forecasting. He is author of over 150 research papers in statistical science and was Editor-in-Chief of the International Journal of Forecasting from 2005 to 2018. In 2007, he received the Moran medal from the Australian Academy of Science for his contributions to statistical research. For over 30 years, Rob has maintained an active consulting practice, assisting hundreds of companies and organisations on forecasting problems.

Thiyanga S. Talagala is a Ph.D. candidate in Statistics at Monash University. Her research focuses on the problem of forecasting large collections of time series. Her reseach interest include Time Series Analysis, Applied Statistics, and Statistical Computing.