International Journal of Forecasting 39 (2023) 641–658

Contents lists available at ScienceDirect

International Journal of Forecasting

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

An accurate and fully-automated ensemble model for weekly time series forecasting

Rakshitha Godahewa [a,*], Christoph Bergmeir [a], Geoffrey I. Webb [a], Pablo Montero-Manso [b]

[a] Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Monash University, Australia
[b] University of Sydney, Australia

ARTICLE INFO

Keywords:
Weekly forecasting
Global models
Ensembling
Meta-learning
Time Series

ABSTRACT

Many businesses and industries require accurate forecasts for weekly time series nowadays. However, the forecasting literature does not currently provide easy-to-use, automatic, reproducible and accurate approaches dedicated to this task. We propose a forecasting method in this domain to fill this gap, leveraging state-of-the-art forecasting techniques, such as forecast combination, meta-learning, and global modelling. We consider different meta-learning architectures, algorithms, and base model pools. Based on all considered model variants, we propose to use a stacking approach with lasso regression which optimally combines the forecasts of four base models: a global Recurrent Neural Network (RNN) model, Theta, Trigonometric Box–Cox ARMA Trend Seasonal (TBATS), and Dynamic Harmonic Regression ARIMA (DHR-ARIMA), as it shows the overall best performance across seven experimental weekly datasets on four evaluation metrics. Our proposed method also consistently outperforms a set of benchmarks and state-of-the-art weekly forecasting models by a considerable margin with statistical significance. Our method can produce the most accurate forecasts, in terms of mean sMAPE, for the M4 weekly dataset among all benchmarks and all original competition participants.

1. Introduction

Even though weekly time series are important in many business contexts, forecasters have traditionally focused on series with lower or higher granularity. The M3 and M4 competitions (Makridakis & Hibon, 2000; Makridakis et al., 2018) were dominated by monthly series, besides strong presences of quarterly and yearly series. On the other hand, daily and sub-daily series also gain considerable attention as they often show multiple seasonalities, which makes them an interesting field of study (Bandara et al., 2019a; O'Hara-Wild & Hyndman, 2018).

Weekly time series generally have only a single seasonality with a long cycle, i.e., a yearly seasonality with a seasonal cycle of approximate length 52.18, which is a large and non-integer value compared with the other commonly considered seasonalities such as quarterly and monthly. On the other hand, when an integer seasonality is required, there are two possible values, 52 and 53, depending on the year. Selection between these integer and non-integer values for the seasonal period depends on the situation. This particular seasonality needs to be properly accounted for in the modelling. Furthermore, due to the long seasonal cycle, weekly series often do not have data spanning over two full periods (two years) where this has consequences for the modelling process.

Even though modelling weekly time series is not as popular a use case as modelling quarterly and monthly series, weekly series are very important in practice. As storing and processing large amounts of data is almost trivial nowadays, and business processes have become more automated, many businesses that have traditionally

* Correspondence to: Monash University, Clayton, Victoria, Australia.
E-mail address: rakshitha.godahewa@monash.edu (R. Godahewa).

https://doi.org/10.1016/j.ijforecast.2022.01.008
0169-2070/

operated on quarterly or monthly bases are now operating on a weekly time scale. Consequently, practitioners are often interested in forecasting product sales weekly.

Consequently, in the M4, the best-performing methods in the weekly category were quite different to the overall winning methods. The method that won the M4-weekly category was a solution based on the commercial Forecast Pro software, discussed in detail by Darin and Stellwagen (2020). Notably, these authors also argue that the M4, in general, is not very representative of a usual business situation, but "the weekly data are perhaps the most similar to business data". Though the method of Darin and Stellwagen (2020) is very accurate, it requires a significant amount of manual intervention and thorough domain knowledge to retrieve good forecasts, and it is based on proprietary software.

On the other hand, the methods that performed well overall in the competition popularised important methodological innovations in the forecasting field. (1) Forecast combination (ensembling) with meta-learning, as used in the second-winning method from Montero-Manso et al. (2020) that combines the forecasts of a set of base algorithms. (2) The use of global forecasting models (Januschowski et al., 2020), as in the winning method from Smyl (2020), which is a global forecasting method (that also uses ensembling at different stages of the algorithm).

The main contribution of this paper is to introduce an accurate, simple, fully-automated, publicly available, and reproducible weekly forecasting method using state-of-the-art forecasting practices of global forecasting models and forecast combinations. In particular, we propose an automated meta-learning ensemble forecasting model for weekly time series forecasting. We analyse different base model pools where the forecasts of the base models are combined using different meta-learning architectures that are based on features, such as the architecture of Montero-Manso et al. (2020), and on stacking (Wolpert, 1992). Furthermore, we use three meta-learning algorithms: linear regression, lasso regression (Tibshirani, 1994), and eXtreme Gradient Boosting (XGBoost, Chen & Guestrin, 2016). Based on all considered model variants, the stacking approach which combines the forecasts of four base models: a globally trained Recurrent Neural Network (RNN, Hewamalage et al., 2021) and three univariate forecasting models: Theta (Assimakopoulos & Nikolopoulos, 2000), Trigonometric Box–Cox ARMA Trend Seasonal (TBATS, Livera et al., 2011), and Dynamic Harmonic Regression Auto-Regressive Integrated Moving Average (DHR-ARIMA, Hyndman & Athanasopoulos, 2018), using lasso regression shows the overall best performance. The sub-models that we use in this best performing proposed method include global and local forecasting models that are especially suitable for weekly forecasting. The most suitable sub-models are automatically selected. Their forecasts are combined using the lasso regression meta-learning algorithm. Hence, in particular, this weekly forecasting method outperforms a set of state-of-the-art weekly forecasting benchmarks with statistical significance across seven experimental datasets on four error metrics. Furthermore, our method produces the most accurate forecasts for the M4 weekly dataset based on the mean of the symmetric Mean Absolute Percentage Error (sMAPE) compared with all original competition participants. Our method is fully-automated, and domain knowledge is not required to obtain the forecasts. Since our method does not use any seasonal features extracted from the series, it is applicable to any weekly dataset irrespective of the series length. This study uses publicly available benchmark datasets representing many real-world applications. However, as their size is limited, when scaling the method to larger datasets, some of our recommendations and modelling choices within the framework, such as lasso regression, may need to be revisited. All implementations related to the method are publicly available at https://github.com/rakshitha123/WeeklyForecasting. Out of our seven benchmark datasets, five datasets are created by aggregating series with higher granularities. We make all the aggregated weekly datasets publicly available for further research use.[1]

The remainder of this paper is organised as follows: Section 2 reviews the related work. Section 3 explains the main components of our analysis, including the details of the base forecasting model pools, meta-learning architectures and algorithms used for aggregating forecasts. Section 4 explains our experimental framework. Section 5 presents the model evaluation results. Finally, Section 6 concludes the paper and discusses possible future research.

2. Related work

In the following, we discuss the related literature in weekly time series forecasting and summarise the state of the art in meta-learning for forecast combination and global modelling.

2.1. Weekly forecasting

The main challenge when forecasting weekly data is the long and non-integer yearly seasonality. The most popular general forecasting techniques Exponential Smoothing State Space Models (ETS, Hyndman, 2008) and Auto-Regressive Integrated Moving Average (ARIMA, Box & Jenkins, 1990) are not able to deal well with such a long seasonal cycle, and hence, their applicability in weekly forecasting is limited (Hyndman & Athanasopoulos, 2018). There are four main ways in the literature to deal with the seasonality in weekly data. The seasonality (1) can be neglected, and a non-seasonal model can be built, it (2) can be addressed with seasonal lags, it (3) can be addressed with seasonal indicator variables such as Fourier terms or seasonal dummy variables, or (4) the series can be deseasonalised before forecasting.

Building non-seasonal models can be a good option if the yearly seasonality is not strong or if less than a full year of data is available. In that case, non-seasonal standard models such as ETS, ARIMA, or Theta (Assimakopoulos & Nikolopoulos, 2000) can be fitted. For example, Padt and Bergmeir (2017) use simple exponential smoothing

[1] https://github.com/rakshitha123/WeeklyForecasting/tree/master/datasets.

in such a situation. Also commonly employed in the literature are other (non-seasonal) non-linear autoregressive models, e.g., from a machine learning domain. Al-qaness et al. (2020) propose an improved version of an Adaptive Neuro-Fuzzy Inference System (ANFIS) to forecast the weekly confirmed influenza cases in China and the USA, which can be used to support health policy-making. Researchers have also proposed models for weekly load forecasting (Barakat & Al-Qasem, 1998), weekly crude oil forecasting (Oussalah & Zaidi, 2018) and weekly groundwater level forecasting (Mohanty et al., 2015) where Neural Networks (NN) and regression models are heavily used with the proposed approaches.

Seasonality can also be modelled by incorporating seasonal lags. For example, Landeras et al. (2009) use seasonal ARIMA and NN models to obtain weekly evapotranspiration forecasts, which are then used in planning and designing water resource systems. However, this approach is hindered by the long seasonal cycle and the non-integer length of the cycle in weekly series. A lag selection mechanism may be necessary to obtain good results.

Fourier terms are also commonly employed nowadays with machine learning models more broadly to model other types of seasonalities (Bandara et al., 2019a). They are commonly used to model the long seasonal cycle of weekly data. TBATS (Livera et al., 2011) and DHR-ARIMA are advocated as suitable methods to be used instead of ETS and ARIMA (Hyndman & Athanasopoulos, 2018) where both methods use Fourier terms to model the seasonality. Pan and Yang (2017) use Autoregressive Moving Average eXogenous (ARMAX) models along with search engine queries, website traffic data and weather information to forecast weekly hotel occupancy for a particular destination. Those authors use Fourier terms and weekly dummy variables to model the seasonality.

Deseasonalisation is another possible approach to handle seasonality. Guttormsen (1999) uses six models to forecast weekly prices for salmon, where the forecasts can then be used to reduce the price fluctuation risks. Here, the time series are deseasonalised before forecasting. That author concludes that Classical Additive Decomposition (CAD) and Vector Auto Regression (VAR) models provide the best price forecasts.

The winning approach of the M4 weekly competition proposed by Darin and Stellwagen (2020) uses a set of experts which is chosen from a pool of baseline models (model families), mostly variations of exponential smoothing and ARIMA, to forecast each series. Furthermore, the baseline models include naïve2, seasonal naïve and dynamic regression models incorporating seasonal lags and Fourier terms to capture the seasonal effects. Short series that seem to be seasonal are customised to capture the seasonal effects properly. One or more models are selected from the pool of baseline models for each series to obtain forecasts based on the series' characteristics. If more models are selected, then the approach uses an out-of-sample testing procedure to select the most appropriate base model to obtain forecasts for a given series. Further investigations are performed to identify the series where the selected experts are not adequate, and finally, the forecasts of such series are modified using domain knowledge as required. Although this model is highly accurate, it requires considerable manual intervention during the forecasting process.

2.2. Meta-learning for forecast combination

Combining forecasts of several heterogeneous models is well-known to improve the forecasting accuracy on average over the individual models (Bates & Granger, 1969; Timmermann, 2006), especially by reducing bias and/or variance (Cerqueira et al., 2017; Wolpert, 1992). In the machine learning space, forecast combination is known as ensembling. Simple averaging is considered as one of the most efficient and accurate forecast combination methods, which tends to be very competitive (Timmermann, 2006). Several weighted averaging combination methods have also been proposed (Cerqueira et al., 2017; Sanchez, 2008).

Recently, more sophisticated models for forecast combinations often use meta-learning approaches. The second winning approach of the M4 forecasting competition, Feature-based Forecast Model Averaging (FFORMA, Montero-Manso et al., 2020), combines the forecasts provided by a set of base models by using an optimal set of weights. These are obtained using a meta-learner trained with series features (for details, see Section 3.2.1). The third winning approach of the M4 forecasting competition proposed by Pawlikowski and Chorowska (2020) also incorporates forecast combinations with meta-learning. The forecasts provided by a set of selected statistical models, including ETS, ARIMA, Theta and Naïve, are optimally combined for each series to obtain the final forecasts. Stacking (Wolpert, 1992) is another widely used meta-learning approach in the forecast combination space (Divina et al., 2018; Khairalla et al., 2018; Zhai & Chen, 2018). It first trains a set of base models using a training set and then uses the predictions of the base models on the training set as inputs to train a meta-learner, which outputs the final predictions. Apart from the base model predictions, the meta-learner can have additional inputs such as features extracted from the series.

2.3. Global models in forecasting

Another relatively recent trend in forecasting, employed in the M4 successfully by the winning solution of Smyl (2020), is the use of *global forecasting models* (Januschowski et al., 2020). Global models build a single model with a set of global parameters across many series. In contrast to local models, they can learn the cross-series information during model training with fewer parameters. This type of models has been pioneered, e.g., by the works of Bandara et al. (2020), Duncan et al. (2001), Flunkert et al. (2017), Montero-Manso and Hyndman (2021), Smyl and Kuber (2016), Trapero et al. (2015). Many researchers have used RNNs in the context of global modelling (Bandara et al., 2019a, 2020, 2019b; Hewamalage et al., 2021; Smyl, 2020).

In this paper, we incorporate the capabilities of different meta-learning architectures that use both global and traditional univariate forecasting models to improve the accuracy of weekly time series forecasting without manual intervention.

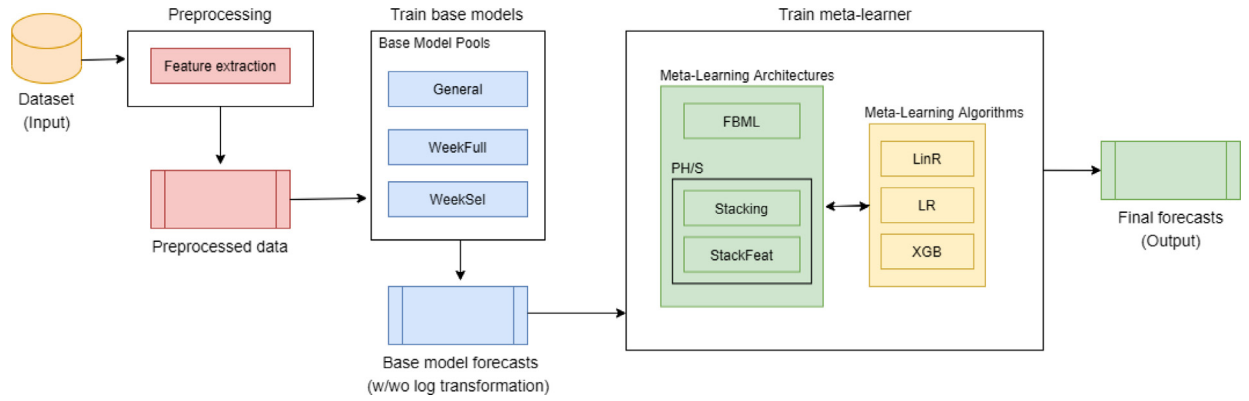

Fig. 1. Overview of our modelling strategy together with the data preprocessing, base model pools, meta-learning architectures and meta-learning algorithms we consider for our study. Here, our goal is to find the best base model pool, meta-learning architecture and meta-learning algorithm to be used as the building blocks of an accurate ensemble weekly forecasting model.

3. Methodology

The main purpose of our paper is to incorporate global modelling, forecast combinations and meta-learning to implement a fully-automated and accurate weekly forecasting model. This section details the base models and the meta-learning architectures, as well as the meta-learning algorithms that we use for aggregating the sub-model forecasts in our analysis. Fig. 1 gives an overview of our modelling strategy together with the data preprocessing, base model pools, meta-learning architectures and meta-learning algorithms we consider for our study.

3.1. Base models

Forecast combination models contain a series of base models where the corresponding base-model forecasts are aggregated to obtain the final forecasts. We consider different base models for our analysis, as detailed in the following sections.

3.1.1. A general base model pool

Forecast combinations are expected to produce better results when the base models are heterogeneous and produce relatively uncorrelated forecasts (de Menezes et al., 2000). Following these guidelines, an excellent starting point for a well-crafted and diverse set of base learners is the FFORMA method (Montero-Manso et al., 2020). FFORMA is the second winning approach of the M4 forecasting competition. It optimally combines forecasts produced by nine sub-forecasting models: ETS, automated ARIMA (Auto.ARIMA), TBATS, Theta, naïve, snaïve, random walk with drift, a locally trained neural network (NNET-AR) and Seasonal and Trend decomposition using Loess with AR modelling of the seasonally adjusted series (STLM-AR). For our analysis, we consider all base models in FFORMA in their implementations in the R package *forecast* (Hyndman & Khandakar, 2008).

We note that FFORMA uses ETS and Auto.ARIMA, two base models that themselves have model selection capabilities, and selects the best model out of various sub-models, using mechanisms such as minimising a bias-corrected version of the Akaike Information Criterion (AICc, Hurvich & Tsai, 1993). Another way of including these base models into a meta-learning framework (also see the approaches of Darin and Stellwagen (2020) and Kolassa (2011)) would be to use all the different model variants as base learners and use the meta-learner to select between the variants. However, while this is a possibility for ETS with a well-defined and relatively low amount of distinct models, for ARIMA models of different orders (or even ARIMA and ETS models with different coefficients), this leads to potentially a very high amount of base models. Therefore, we follow the approach of FFORMA and opt for using any model selection procedures present in the base learners, under the assumption that they are crafted with expert knowledge of the base learners and that this leads to a model pool of few strong and heterogeneous base models.

3.1.2. Base models for weekly forecasting

Some models are more suitable for weekly forecasting than others because of the general base model pool identified above. The simple non-seasonal models Theta, naïve, and random walk with drift have limited suitability, as they do not consider the seasonality. If they have a trend component, they will instead model the seasonality as a trend, leading to very bad accuracies. However, it may not be possible to model the seasonality in practice due to the long seasonality period. If the seasonal pattern is not strong or slow-moving, ignoring it or modelling it through a trend may be acceptable. In the literature (Section 2.1), there are many examples of non-seasonal models applied to weekly series with good results. Next, snaïve and STLM-AR are simple methods that can model the seasonality in weekly data, though STLM-AR needs two full periods. TBATS (Livera et al., 2011) can deal with long and non-integer periodic effects of time series, which makes it one of the state-of-the-art methods used in weekly time series forecasting (Hyndman & Athanasopoulos, 2018). TBATS models the seasonality using a Fourier-series-based trigonometric representation. ETS (Hyndman, 2008), as implemented in the `ets()` function in the *forecast* package in R (Hyndman & Khandakar, 2008), in the weekly forecasting context is a non-seasonal

model as it does not handle seasonal periods greater than 24. A different implementation of ETS is available in the R package *smooth* (Svetunkov, 2020). In contrast to ETS, this implementation handles seasonal cycles greater than 24 and hence, it can be used as a seasonal model in the weekly context. However, Hyndman (2008) argues that exponential smoothing has limitations if used with long seasonal cycles, and the limitation of `ets()` to handle seasonal cycles only up to the length 24 is a design choice and not a limitation of the implementation. In line with this, we performed preliminary experiments with seasonal ETS as a base model, which did not provide promising results and confirmed this assumption. Hence, we do not consider seasonal ETS and use ETS only in a non-seasonal version. Auto.ARIMA (Hyndman & Khandakar, 2008) is a state-of-the-art statistical benchmark; however, again, due to the long seasonal cycle of weekly series, it is considered not adequate for weekly series (Hyndman & Athanasopoulos, 2018). With DHR-ARIMA, an alternative better suited for weekly time series exists, discussed in the following. Similarly, NNET-AR is a locally trained feed-forward NN, and much better alternatives exist; namely, we will use globally trained RNNs, also discussed in the following.

Dynamic Harmonic Regression ARIMA. This technique is especially suitable to forecast series with long seasonal periods such as weekly series (Hyndman & Athanasopoulos, 2018). The seasonality is fixed during modelling. Here, Fourier terms and ARMA errors model seasonality and short-term time series dynamics.

Fourier terms are a set of sine and cosine pairs that are useful in modelling periodic effects in time series (Harvey & Shephard, 1993), especially to model the long seasonal periods presented in weekly data. The Fourier terms related to a particular time point of a series can be obtained using the formula shown in Eq. (1), where t is the time point, s is the seasonal periodicity of the time series, and k is the number of sine cosine pairs used with the transformation.

$$sin\left(\frac{2\pi kt}{s}\right), cos\left(\frac{2\pi kt}{s}\right) \tag{1}$$

The number of Fourier terms controls the smoothness of the seasonal pattern. We find the optimal number of Fourier terms to be considered with a particular set of series using a grid-search approach considering the range from 1 to 25.

We use the *auto.arima* function from the *forecast* package (Hyndman et al., 2020) along with the Fourier terms generated by using the *fourier* function to obtain DHR-ARIMA forecasts for a given weekly series.

Recurrent Neural Networks with Long Short-Term Memory Cells. An RNN is a type of NN that is especially suitable for sequence modelling problems (Elman, 1990) due to feedback loops in its architecture. The winning approach of the M4 forecasting competition (Smyl, 2020) in 2018 was based on RNNs, which has given them considerable attention in the forecasting community lately. We include this model due to heterogeneity considerations. Combining the forecasts of a globally trained RNN with the forecasts obtained from the other locally trained univariate forecasting models incorporates the strengths of both global and local models while mitigating the weaknesses of each other (Section 5). Extracting seasonal patterns through globality is another benefit of using a global RNN (Montero-Manso & Hyndman, 2021).

We use a Long Short-Term Memory (LSTM) with peephole connections (Gers et al., 2000), following the methodology proposed by Hewamalage et al. (2021). As preprocessing steps, series are normalised by dividing them by corresponding mean values. Then, the normalised series are transformed to a log scale for variance stabilisation.

We use two techniques to deal with periodic/seasonality effects in the time series. If the corresponding timestamps are available with each data point in all series and if it is possible to align series across time, then we use Fourier terms (Hyndman & Athanasopoulos, 2018) to capture the periodic effects of the weekly time series, where a set of sine and cosine terms are used together with the preprocessed time series when training the RNN model. (Bandara et al., 2019a). The *fourier* function in the R package *forecast* (Hyndman et al., 2020) is used for Fourier term calculations. Suppose the corresponding timestamps are not available, or it is not possible to align series across time, which is the case, e.g., in the M4 weekly dataset (Makridakis et al., 2018). In that case, we use a seasonal lag such as 53 to train the RNN model. Our RNN model uses a stacked architecture (Bandara et al., 2019b; Godahewa et al., 2020; Hewamalage et al., 2021), with input and output windows for multi-step-ahead forecasting (Hewamalage et al., 2021).

Eight hyperparameters need to be chosen during our RNN model training, namely: cell dimension, mini-batch size, the maximum number of epochs, epoch size, number of hidden layers, L2-regularisation weight, standard deviation of random normal initialiser and standard deviation of the Gaussian noise. We perform automatic hyperparameter tuning using the Sequential Model-based Algorithm Configuration (SMAC) optimisation method (Hutter et al., 2011) as described by Hewamalage et al. (2021). The COntinuous COin Betting (COCOB, Orabona & Tommasi, 2017) algorithm is used as the learning algorithm due to its capability of selecting an optimal learning rate during model training.

3.1.3. Base model pools for analysis

We have discussed a general base model pool and a base model pool developed for weekly series. However, following the idea of a parsimonious, heterogeneous base model pool to produce uncorrelated base model forecasts, we further select a subset of the most suitable base models that are diverse as well as accurate to make the forecast combination model stronger (Chandra et al., 2006).

TBATS and DHR-ARIMA are currently considered state-of-the-art in weekly forecasting, as they can capture seasonal patterns in such series. We include them in the selection, as well as the RNN, as we have specifically crafted it for weekly data, and it is a global forecasting model able to extract patterns across series.

We do not consider straightforward base models for this model pool and therefore omit the snaïve method.

STLM-AR is also omitted as it requires two full periods of data. As discussed in Section 3.1.2, non-seasonal models are unsuitable for weekly data in principle but are still often used and lead to good results. Thus, we select only one model out of the non-seasonal models, namely the Theta method. Theta models the trend in a series as half the slope of a linear regression of the data (Hyndman & Billah, 2003). Thus, it models the trend conservatively and may be better suited for cases where a yearly seasonality in the data is modelled through a trend component. Furthermore, Theta is simple but highly competitive, having been the winning method of the M3 forecasting competition (Makridakis & Hibon, 2000).

Thus, we use the following three base model pools for our analysis.

General This model pool consists of the nine base models used in FFORMA, not focused on forecasting of weekly series in particular: ETS, Auto.ARIMA, TBATS, Theta, naïve, snaïve, random walk with drift, NNET-AR and STLM-AR.

WeekFull Contains DHR-ARIMA, RNN and seven base models used in FFORMA: ETS, TBATS, Theta, naïve, snaïve, random walk with drift and STLM-AR.

WeekSel Contains the selected four diverse base models suitable for weekly forecasting and highly accurate: TBATS, DHR-ARIMA, Theta and a global RNN.

Orthogonally to the choice of base models, we also analyse the effect of applying a logarithmic transformation to the base model forecasts, which is a common preprocessing technique used in the forecasting space to stabilise the variance of forecasts (Bandara et al., 2020; Smyl, 2020). We apply a logarithm to the base model forecasts before using them with the forecast combination approaches. All time series in our experimental datasets are non-negative. If the base model forecasts contain zeros, we add a constant $c = 1$ before transforming them with the logarithm.

3.2. Meta-learning architectures

We explore the use of a feature-based meta-learning approach and stacking as two meta-learning architectures for weekly forecasting, as discussed in the following.

3.2.1. Feature-based meta-learning

We refer to the meta-learning architecture used in the FFORMA (Montero-Manso et al., 2020) algorithm as feature-based meta-learning (FBML) in the remainder of the paper. It optimally combines the forecasts from the base models using a set of weights obtained by a gradient boosted tree with 42 input features. The input features are calculated from the original time series, and they include trend, seasonality, linearity, curvature, and correlation features calculated using the R package *tsfeatures* (Hyndman et al., 2019).

In the training phase, the algorithm splits each series into a training period, and a validation period taken from the end of the series with a length equal to the forecasting horizon. The features are calculated using the training period. Then, for each base model, the forecasts and the respective errors are calculated for the validation periods of all series. A gradient boosted tree is then trained as a meta-learning model over all series to find a function that maps the features to a set of optimal weights that can be used to combine the base model forecasts in a way that the total loss over the validation periods gets minimised.

The features are calculated for the full series in the prediction phase, and base model forecasts are obtained for the expected forecast horizon. Finally, the base model forecasts are combined using the optimal set of weights provided by the meta-learner given the calculated features.

Some drawbacks of this approach are that it generally requires a large number of long series to learn the sub-model weights. Furthermore, the lengths of the series should be at least two seasonal cycles to calculate the seasonal features, which is often a difficult requirement for weekly series due to the long seasonal cycle. With our experimental datasets, most of the seasonal features cannot be calculated for short series with less than two full periods and hence for those, only non-seasonal features are calculated. Simpler meta-learning architectures such as stacking can be considered good alternatives under these circumstances, as discussed in the following section.

3.2.2. Stacking

Stacking (Wolpert, 1992) is a widely used meta-learning architecture and a good option to combine the forecasts of different base model pools. Stacking trains a meta-model by using the forecasts provided by the base models as inputs, and it directly outputs the final forecasts. In the stacking approach, we consider the last sequence of values in a given time series equal to the size of the forecast horizon as its validation part. The base model forecasts are separately computed for this validation part, and a meta-learner is trained with them considering the validation part as the true output. Then, the base model forecasts are separately calculated using the whole time series corresponding with the actual test period. The base model forecasts are provided as the test inputs, and the final forecasts corresponding with the test period are obtained using the previously trained meta-learning model. Compared with the FBML approach, stacking is capable of learning the sub-model weights with smaller amounts of data. If enough data to calculate the features is available, stacking can also be done in a way where both sub-model forecasts and features are used as inputs of the stacked meta-learning model. In the remainder of the paper, we name this strategy StackFeat. In this stacking variant, we use the sub-model forecasts as well as the 42 features used in FBML as inputs during model training. We also analyse whether training one meta-learner per horizon or training a single global meta-learner across all horizons is the best approach in the weekly forecasting context. When using a single meta-learner, we convert the sub-model forecasts into a vector format before training the meta-learner. In the remainder of the paper, we use PH to denote training multiple meta-learners per horizon and S to denote training a single global meta-learner across all horizons.

3.3. Meta-learning algorithms

For our analysis, we use three algorithms for meta-learning, namely XGBoost (Chen & Guestrin, 2016), linear regression, and lasso regression (Tibshirani, 1994). XGBoost is a state-of-the-art machine learning algorithm usually leading to very good accuracy, and it is the meta-learner used in FFORMA, which are the reasons for us to use it as a meta-learner in this work. We use the *xgboost* function in the R package *xgboost* (Chen et al., 2020) to implement the XGBoost meta-learner. The hyperparameters of the XGBoost meta-learner, such as η and maximum tree depth, are tuned using a grid search approach. We also consider simpler meta-learning algorithms suitable for our analysis with relatively small datasets, as they are common in forecasting weekly data. Linear regression is a simple meta-learner that combines a given set of forecasts. A linear model is fitted using the sub-model forecasts, and the forecasts are combined based on their corresponding coefficient values. We use the *glm* function in the R package *glmnet* (Friedman et al., 2010) to implement the linear regression meta-learner. Model combination generally can be considered more accurate than model selection (Kolassa, 2011), and model selection can be seen as a special case of model combination. For our analysis, we also consider lasso regression as a meta-learning algorithm that performs both model combination and selection, excluding the worst-performing models from the combinations. Lasso regression prevents model overfitting that can occur with the normal linear regression by adding a regularisation term (L1) to the cost function of linear regression. Hence, the coefficients corresponding with the worst-performing models shrink towards zero. Prior works of using local lasso regression models for combining forecasts have shown good results (Diebold & Shin, 2019; Wilms et al., 2016). We use the *glmnet* function in the R package *glmnet* (Friedman et al., 2010) to implement the lasso regression model. The regularisation parameter λ is tuned as a hyperparameter with 10-fold cross-validation. In the remainder of the paper, we use LR, LinR, and XGB to denote lasso regression, linear regression, and XGBoost, respectively.

4. Experimental framework and results

This section presents our experimental setup and results on seven benchmark datasets for the weekly forecasting models.

4.1. Datasets

We use seven publicly available datasets to evaluate the performance of our proposed weekly forecasting models. Two datasets initially contain weekly series, and the series in the remaining five datasets are aggregated from lower granularities accordingly to make them weekly. Table 1 provides a summary of the datasets in their weekly aggregated versions used in our work. A brief overview of the datasets is as follows.

- M4 Weekly Dataset: The weekly dataset of the M4 forecasting competition (Makridakis et al., 2018).
- NN5 Dataset: The dataset of the NN5 forecasting competition contains 111 daily time series of daily cash withdrawals from Automatic Teller Machines (ATM) in the UK (Taieb et al., 2012). The dataset has missing values, which we replace with the median before the temporal aggregation.
- Reduced Kaggle Wikipedia Web Traffic Dataset: We use the first 1000 time series from the Kaggle Wikipedia Web Traffic forecasting competition (Google, 2017). The time series show the number of hits/traffic for a given set of Wikipedia web pages per day. We replace missing values of the dataset with zeros before aggregation.
- Ausgrid Energy Dataset: A half-hourly dataset that contains 300 time series representing the general energy consumption of Australian households (AusGrid, 2019). One series was omitted before aggregation as it has missing values for more than eight consecutive months.
- San Francisco Traffic Dataset: An hourly dataset that contains 862 time series representing the road occupancy rates on San Francisco Bay area freeways from 2015 to 2016 (Caltrans, 2020; Lai, 2017).
- Solar Dataset: A dataset that contains 137 time series representing the solar power production records per every 10 min in the state of Alabama in 2006 (Lai, 2017; Solar, 2020).
- Dominick Dataset: A weekly dataset showing the sales of 28 product categories of Dominick's Finer Foods, a large American retail chain in the Chicago area (Center, 2020).

Though many of the datasets have been aggregated from smaller granularities to weekly, we argue that there is no difference between "pure" weekly datasets and datasets aggregated into weekly datasets from, e.g., daily data. Take as an example sales data in retail. Those will be Point of Sale (POS) data that are time stamped with their exact transaction time throughout the day. As such, daily sales are as much an aggregation as are weekly sales. Taking daily sales and aggregating them further up to weekly sales is no different than aggregating the transaction data directly to a weekly dataset. This consideration will hold for any timestamped transaction data, like ATM withdrawals, ride-share rides, daily webpage hits, and others. This consideration is also valid to electricity and power production data, where the weekly electricity usage and power production can be determined by aggregating the corresponding lower granularity data, such as hourly/minutely observations.

4.2. Error metrics

We use two error measures that are common in the forecasting research space: sMAPE and Mean Absolute Scaled Error (MASE, Hyndman & Koehler, 2006), to measure the performance of our models. They are defined in Eqs. (2) and (3), where M is the number of instances in the training set, S is the length of the seasonal cycle of

Table 1
Summary of the Used Datasets.

Dataset	No. of series	Forecast horizon	Min. length	Max. length
M4	359	13	80	2597
NN5	111	8	105	105
Web Traffic	1000	8	106	106
Ausgrid	299	8	148	148
Traffic	862	8	96	96
Solar	137	5	44	44
Dominick	28	8	208	391

the dataset, h is the forecast horizon, F_k are the generated forecasts, and Y_k are the actual values.

$$sMAPE = \frac{100\%}{h} \sum_{k=1}^{h} \frac{|F_k - Y_k|}{(|Y_k| + |F_k|)/2} \tag{2}$$

$$MASE = \frac{\sum_{k=M+1}^{M+h} |F_k - Y_k|}{\frac{h}{M-S} \sum_{k=S+1}^{M} |Y_k - Y_{k-S}|} \tag{3}$$

For datasets containing zeros, namely the Kaggle web traffic and San Francisco traffic datasets in our experiments, we use the variant of the sMAPE proposed by Suilin (2017), which eliminates problems with small values and division by zero by changing $(|Y_k|+|F_k|)$ in the denominator of the sMAPE as defined in Eq. (2) to $max(|Y_k| + |F_k| + \epsilon, 0.5+\epsilon)$. We set the parameter ϵ to its proposed default of 0.1.

The MASE measures the performance of a model compared with the in-sample average performance of a one-step-ahead naïve or snaïve benchmark. For the M4 dataset, we calculate the MASE using the naïve benchmark to compare our results with the original competition results. For the remaining datasets, we choose the snaïve or naïve benchmarks depending on the series length. If the dataset is seasonal and has at least one full cycle of data points (52 data points for weekly data), then we calculate the MASE using the snaïve benchmark, otherwise using the naïve benchmark. In particular, the maximum series length of the Solar dataset is 44, so we use the naïve benchmark here. The remaining five datasets have series with more than one full cycle of data points, and the snaïve benchmark is used.

To measure the performance of the models across series, we further calculate the mean and median values of sMAPE and MASE across the series. Thus, each model is evaluated using four error metrics: mean sMAPE, median sMAPE, mean MASE, and median MASE across a dataset.

4.3. Benchmarks and variants

We use the eleven base models: RNN, DHR-ARIMA, ETS, Auto.ARIMA, TBATS, Theta, naïve, snaïve, random walk with drift, NNET-AR and STLM-AR as the main benchmarks. Furthermore, the averages of the forecasts provided by the base model pools are also considered as benchmarks. We name these benchmarks according to their model pool Average_General, Average_WeekFull, and Average_WeekSel, respectively.

Based on the base model pools, meta-learning architectures, meta-learning algorithms, number of meta-learners, and the incorporation of features and the logarithmic transformation explained in Section 3, 90 different model variants can be considered for the experiments. To reduce the number of models to run, make the experiments computationally feasible, and reduce the risk of spurious results, we first perform experiments to determine the best model pool to use and then focus on this model pool afterwards. Also, as the prediction task is more complex with the FBML approach as it predicts instead of a single forecast a number of weights for the sub-models, the FBML approach is only tested with its proposed meta-learning algorithm, XGB, and we do not consider the more straightforward lasso regression and linear regression as a meta-learner in this case. We always build one global meta-learner with the FBML approach, and logarithmic transformation is not considered, as proposed in the original FFORMA method. Thus, out of the 90 possible model variants, 26 are finally compared to identify the best model to forecast the weekly series, as detailed in the results section.

4.4. Statistical tests of the results

We use the non-parametric Friedman rank-sum test to assess the statistical significance of the results provided by different forecasting methods across time series considering a significance level of $\alpha = 0.05$. The methods are ranked on every series of all datasets, namely M4, NN5, Kaggle web traffic, Ausgrid, Traffic, Solar and Dominick, based on their corresponding sMAPE errors. As this procedure gives the same weight to every series in the comparison, datasets with larger amounts of series naturally get a higher weighting. We deem this not a problem in our analysis as the datasets used have comparable sizes. According to the average rank, the best method is chosen as the control method. Furthermore, Hochberg's posthoc procedure is used to further characterise the statistical differences (García et al., 2010).

5. Results and discussion

This section details the results in terms of main accuracy and statistical significance and later also gives some results that provide more insights into the modelling.

Table 2
sMAPE results for methods that perform best overall and per category across all experimental datasets. The results of all model variants across all error metrics are available in the Online Appendix. The best performing models in each group are italicised and the overall best performing models are highlighted in boldface.

	M4	NN5	Kaggle	Ausgrid	Traffic	Solar	Dominick
Mean sMAPE							
Theta	8.70	12.06	30.42	29.99	12.48	24.76	15.86
TBATS	7.14	11.63	31.48	25.14	12.81	***18.93***	16.95
DHR-ARIMA	8.82	11.32	42.81	25.63	13.58	20.38	16.88
RNN	7.77	***10.38***	26.86	25.56	12.24	27.78	15.23
ETS	8.86	12.22	29.84	30.01	12.44	23.40	15.86
Naïve	9.16	13.27	31.35	31.23	12.98	32.80	20.50
Snaïve	13.94	16.48	46.96	30.17	19.42	19.94	22.82
Random Walk	9.48	13.27	31.99	33.35	13.15	34.38	20.63
STLM-AR	9.21	12.45	43.71	27.05	12.73	27.37	17.30
NNET-AR	9.45	14.52	39.69	26.62	14.35	27.56	17.19
Auto.ARIMA	8.67	13.50	32.46	26.59	12.66	25.65	***14.71***
Average_General	6.93	10.94	30.10	23.29	12.53	25.25	16.10
Average_WeekFull	***6.78***	10.81	30.02	22.99	12.27	24.71	16.08
Average_WeekSel	6.84	***10.38***	***29.58***	**22.66**	***11.67***	22.81	15.39
Stack_LR_S_Log_WeekFull	6.69	10.32	**26.73**	***23.02***	12.21	22.48	16.81
Stack_LinR_S_Log_WeekFull	6.98	10.41	27.05	23.67	12.20	**15.67**	16.95
Stack_LR_S_Log_WeekSel	**6.42**	**10.16**	**26.73**	24.45	11.76	19.02	***14.61***
Stack_LinR_S_Log_WeekSel	6.71	10.17	26.93	24.10	***11.63***	19.18	15.57
Stack_LR_PH_Log_WeekSel	6.93	12.04	***26.82***	25.85	11.26	20.57	17.10
Stack_LinR_PH_Log_WeekSel	6.94	12.13	27.06	24.94	**11.05**	21.49	17.61
StackFeat_LR_S_WeekSel	7.07	***10.55***	27.19	***22.87***	11.99	18.92	24.98
StackFeat_XGB_S_WeekSel	8.47	13.32	34.38	30.85	13.74	***16.92***	30.20
StackFeat_LR_S_Log_WeekSel	***6.87***	10.65	27.27	25.34	11.96	18.77	***16.98***
FBML_XGB_S_General	7.91	13.34	34.30	26.45	12.41	24.98	18.19
FBML_XGB_S_WeekSel	***6.96***	***10.66***	***32.43***	***24.46***	***12.01***	***23.50***	**14.36**
Median sMAPE							
Theta	5.40	10.97	26.20	24.31	9.72	24.90	13.31
TBATS	4.94	11.15	27.28	17.83	10.00	***18.02***	14.51
DHR-ARIMA	5.12	11.07	34.34	18.98	***8.39***	19.60	14.04
RNN	5.09	9.94	***24.25***	19.15	9.60	27.69	***12.55***
ETS	5.15	10.85	25.91	24.82	9.77	24.51	13.76
Naïve	5.18	10.89	27.49	26.28	10.20	32.07	15.67
Snaïve	8.36	15.40	40.01	21.96	14.83	20.38	20.69
Random Walk	5.16	10.99	27.46	27.87	10.30	33.90	15.62
STLM-AR	6.46	11.16	37.99	21.39	10.27	27.28	15.06
NNET-AR	5.05	12.83	31.70	19.98	9.68	27.19	14.39
Auto.ARIMA	5.25	12.22	26.48	20.06	10.01	25.68	12.71
Average_General	4.73	10.02	26.31	17.10	9.72	25.13	14.10
Average_WeekFull	4.84	***9.78***	26.32	17.27	9.54	24.79	14.46
Average_WeekSel	***4.51***	9.86	25.61	**16.13**	8.75	22.54	14.14
Stack_XGB_S_Log_General	6.38	13.77	30.77	23.20	11.25	21.62	**12.44**
Stack_LR_S_Log_WeekFull	4.62	9.83	24.30	***16.74***	9.60	22.00	13.88
Stack_LR_S_Log_WeekSel	**4.08**	9.56	**24.13**	18.26	9.03	19.03	13.10
Stack_LinR_S_Log_WeekSel	4.40	**9.49**	24.38	17.04	***8.83***	19.46	13.43
Stack_XGB_S_Log_WeekSel	6.15	12.48	30.32	26.05	11.62	**15.99**	19.02
Stack_LR_PH_Log_WeekSel	4.60	11.52	***24.42***	19.37	8.31	19.66	14.73
Stack_LinR_PH_Log_WeekSel	4.58	11.63	24.63	18.41	**8.02**	21.24	***13.83***
StackFeat_LR_S_WeekSel	***4.28***	***9.70***	24.69	***16.62***	9.16	18.91	15.41
StackFeat_XGB_S_Log_WeekSel	6.55	12.49	28.83	23.06	10.58	***16.13***	16.28
FBML_XGB_S_General	***4.52***	11.92	27.93	19.78	9.63	25.01	13.23
FBML_XGB_S_WeekSel	4.69	***10.10***	***26.46***	***17.01***	***9.50***	***23.54***	***12.61***

5.1. Main accuracy results

Tables 2 and 3 present the results of a selected set of model variants and benchmarks across all experimental datasets for mean/median sMAPE and mean/median MASE, respectively. The results of all model variants across all error metrics are available in an Online Appendix.[2]

[2] The Online Appendix is available at https://doi.org/10.1016/j.ijforecast.2022.01.008.

Table 3
MASE results for methods that perform best overall and per category across all experimental datasets. The best performing models in each group are italicised and the overall best performing models are highlighted in boldface.

	M4	NN5	Kaggle	Ausgrid	Traffic	Solar	Dominick
Mean MASE							
Theta	2.734	0.898	0.688	1.206	1.122	1.224	0.877
TBATS	2.249	0.864	0.692	1.045	1.149	***0.910***	0.936
DHR-ARIMA	2.418	0.854	0.834	1.068	1.023	0.991	0.926
RNN	2.482	***0.768***	***0.608***	1.042	1.128	1.422	***0.811***
ETS	2.334	0.902	0.685	1.209	1.122	1.158	0.870
Naïve	2.777	0.974	0.738	1.258	1.178	1.735	1.253
Snaïve	9.736	1.160	1.016	1.229	1.581	0.942	1.197
Random Walk	2.682	0.977	0.751	1.316	1.191	1.842	1.262
STLM-AR	3.519	0.915	0.884	1.150	1.135	1.382	0.958
NNET-AR	3.015	1.099	1.236	1.090	1.152	1.410	0.988
Auto.ARIMA	2.377	1.000	0.696	1.134	1.131	1.285	0.829
Average_General	2.742	0.812	0.704	0.952	1.110	1.257	0.893
Average_WeekFull	2.698	0.802	0.663	0.941	1.089	1.227	0.893
Average_WeekSel	***2.142***	0.777	0.653	**0.930**	***1.020***	1.119	0.852
Stack_LR_S_Log_WeekFull	2.227	0.772	0.607	***0.945***	1.112	1.098	0.932
Stack_LinR_S_Log_WeekFull	2.208	0.774	0.612	0.965	1.117	**0.732**	0.921
Stack_LR_S_Log_WeekSel	**2.112**	**0.761**	**0.606**	1.004	1.053	0.905	***0.802***
Stack_LinR_S_Log_WeekSel	2.281	0.763	0.610	0.982	***1.039***	0.926	0.857
Stack_LR_PH_Log_WeekSel	2.599	0.909	***0.608***	1.074	1.024	1.013	0.957
Stack_LinR_PH_Log_WeekSel	2.637	0.918	0.612	1.028	**1.004**	0.989	0.958
StackFeat_LR_PH_Log_WeekSel	2.434	0.941	0.616	1.080	1.064	1.126	1.041
StackFeat_LR_S_WeekSel	2.340	***0.784***	0.617	***0.948***	1.111	0.901	1.181
StackFeat_LR_S_Log_WeekSel	***2.259***	0.794	0.617	1.051	1.108	0.895	***0.941***
StackFeat_XGB_S_Log_WeekSel	4.002	0.954	0.728	1.195	1.264	***0.755***	1.316
FBML_XGB_S_General	***2.156***	0.944	0.723	1.062	1.683	1.249	1.097
FBML_XGB_S_WeekSel	2.194	***0.796***	***0.659***	***0.994***	***1.541***	***1.159***	**0.789**
Median MASE							
Theta	1.969	0.805	0.549	0.981	0.983	1.241	0.843
TBATS	**1.414**	0.874	0.553	0.848	0.997	***0.892***	0.795
DHR-ARIMA	1.640	0.826	0.686	0.903	**0.793**	0.967	0.778
RNN	1.582	**0.680**	**0.483**	0.867	0.924	1.426	***0.697***
ETS	1.801	0.765	0.534	0.992	0.978	1.227	0.856
Naïve	1.938	0.776	0.556	1.046	1.023	1.707	1.022
Snaïve	3.099	1.115	0.838	1.026	1.442	0.974	1.156
Random Walk	1.864	0.822	0.557	1.076	1.035	1.824	1.023
STLM-AR	2.053	0.867	0.738	0.986	0.953	1.400	0.857
NNET-AR	1.963	0.976	0.678	0.928	0.924	1.350	0.868
Auto.ARIMA	1.660	0.936	0.547	0.905	0.997	1.289	0.843
Average_General	1.680	0.749	0.537	0.753	0.956	1.269	0.815
Average_WeekFull	1.674	0.703	0.531	**0.724**	0.930	1.234	0.810
Average_WeekSel	1.514	0.730	0.516	0.730	0.850	1.114	0.704
Stack_LR_S_Log_WeekFull	1.483	***0.690***	***0.485***	0.756	0.953	1.115	0.808
Stack_LinR_S_Log_WeekFull	***1.423***	0.726	0.488	***0.752***	0.942	**0.749**	0.824
Stack_LR_S_Log_WeekSel	1.512	0.691	***0.485***	0.799	0.879	0.936	***0.712***
Stack_LinR_S_Log_WeekSel	1.497	0.694	0.487	0.777	***0.848***	0.960	0.770
Stack_LR_PH_Log_WeekSel	1.836	0.841	***0.484***	0.891	0.816	0.980	0.909
Stack_LinR_PH_Log_WeekSel	1.793	0.846	0.485	0.843	***0.798***	0.986	***0.815***
StackFeat_LR_S_WeekSel	1.607	0.753	0.501	***0.726***	0.891	0.929	1.026
StackFeat_LR_S_Log_WeekSel	***1.492***	***0.739***	0.496	0.819	0.888	0.922	0.886
StackFeat_XGB_S_Log_WeekSel	2.054	0.918	0.582	0.979	1.018	***0.755***	0.959
FBML_XGB_S_General	1.569	0.899	0.553	0.874	1.372	1.272	0.823
FBML_XGB_S_WeekSel	***1.560***	***0.696***	***0.528***	***0.775***	***1.125***	***1.155***	**0.644**

5.1.1. Naming of model combinations

The model combinations in Tables 2 and 3, and Tables A.1 and A.2 in the Appendix are named in a way that they explain the corresponding used base model pool, meta-learning architecture, meta-learning algorithm, number of meta-learners, and whether log transformation is applied as a preprocessing technique. In particular, the model combination names have the following structure.

$\langle architecture \rangle\_\langle algorithm \rangle\_\langle number-mls \rangle\_\langle log \rangle\_\langle model-pool \rangle$

The meta-learning architecture (Section 3.2) is one of FBML, Stack or StackFeat (stacking with features). The meta-learning algorithm (Section 3.3) is one of lasso regression (LR), linear regression (LinR) or XGBoost (XGB). The number of meta-learners (Section 3.2.2) can be either one meta-learner per horizon (PH) or single meta-learner

(S). The base model pool (Section 3.1) is one of General, WeekFull or WeekSel.

5.1.2. Grouping of sub-experiments

The models in Tables 2 and 3, and Tables A.1 and A.2 in the Appendix are grouped based on the sub-experiments. The results of the best-performing models in each group are italicised. The overall best performing models across the datasets are highlighted in bold. We see that in the first group, which contains the benchmark models, the Average_WeekSel method performs best for all datasets except the M4, Solar and Dominick datasets. On the M4 dataset, Average_WeekFull performs the best in the first group. On the Solar dataset, TBATS and Snaïve perform better. On the Dominick dataset, Auto.ARIMA and RNN perform better.

These results already indicate that the WeekSel model pool may be the most suitable model pool for weekly time series forecasting. In the next experiment, we focus on which base model pool is the most suitable.

To make the experiments more tractable, we fix the meta-learning architecture in the following experiment. We assume that the meta-learning architecture will have little influence on which base-model pool works well, compared with the choice of meta-learning algorithms, which will have a larger influence. For example, the lasso regression meta-learning algorithm has the capability to drop sub-models when their contribution is negligible to the final forecasts. Therefore, we assume some meta-learning algorithms can work better with larger model pools than others. We consider a model configuration that uses the Stack architecture and trains a single global meta-learner across all horizons (S). By training a single meta-learner, it is the most straightforward architecture among all architectures considered in our work. Furthermore, we use the log-transformed sub-model forecasts. From preliminary experiments not reported here, we observe that training the model with log-transformed forecasts provides better results for the experimental datasets than the models that use forecasts in the original scale. This simple model configuration is tested with all three base model pools (General, WeekFull, WeekSel) combined with three meta-learning algorithms (LR, LinR, XGB), making nine different model combinations.

The second group of models in Tables A.1 and A.2 in the Appendix shows that the variants generally provide better results with the WeekSel model pool across the experimental datasets and meta-learning algorithms. Hence, we limit the subsequent experiments to this model pool, discarding the other two model pools.

In the third experiment, we further analyse the usage of incorporating features during the training of stacking models (StackFeat), the effect of applying log transformation for sub-model forecasts and training multiple meta-learners one per horizon (PH) in contrast to training a single global meta-learner. All possible modelling combinations of StackFeat are executed with the best base model pool, WeekSel, as shown in Tables A.1 and A.2 in the Appendix. The variants that use lasso regression generally show better performance in this group. The StackFeat architecture also performs better when it builds a single meta-learner across all horizons. The usage of the log-transformed forecasts shows a mixed performance with the StackFeat architecture-based variants.

In Tables 2 and 3, and Tables A.1 and A.2 in the Appendix, in the last group of models, FBML_XGB_S_General refers to the original FFORMA approach. As explained in Section 4.3, the FBML approach is only tested under its proposed settings by only building one global meta-learner with the XGBoost meta-learning algorithm without considering logarithmic transformation. The FBML approach is also tested with the best model pool, WeekSel, and we see that using this model pool leads to consistently improved results.

5.1.3. Best-performing model variant: Stack_LR_S_Log_WeekSel

Based on all considered variants shown in Tables A.1 and A.2 in the Appendix, Stack_LR_S_Log_WeekSel overall provides the best forecasts. We see that on mean sMAPE, Stack_LR_S_Log_WeekSel outperforms all base models for all experimental datasets except for the Solar dataset. Furthermore, it outperforms all benchmarks and variants on the M4, NN5 and Kaggle web traffic datasets. For the M4 weekly dataset, Stack_LR_S_Log_WeekSel produces a mean sMAPE of 6.42, which places it at the first position in the original results of the M4 weekly competition (Mcompetitions, 2018), with an sMAPE result of 6.58 of the original top solution.

Overall, Stack_LR_S_Log_WeekSel demonstrates a better performance on median sMAPE than the other benchmark models and variants where it is the best performing model on M4 and Kaggle web traffic datasets.

On mean MASE, Stack_LR_S_Log_WeekSel outperforms the individual base models for all experimental datasets except the Traffic dataset. Furthermore, it is the best-performing model on the M4, NN5, and Kaggle web traffic datasets. For the M4 weekly dataset, Stack_LR_S_Log_WeekSel produces a mean MASE of 2.112, which places it at the third position in the original results of the M4 weekly competition. In the initial results, FFORMA was at the second position with a mean MASE of 2.108. A difference between the original participating solution of FFORMA in the M4 competition and FBML_XGB_S_General in our experiments is that those authors use the full M4 dataset containing 100,000 series for model training. But in our case, as our focus is on weekly series, we only use the 359 weekly series in the M4 dataset for FBML_XGB_S_General model training, which produces a mean MASE of 2.156. Thus, FBML_XGB_S_General in this configuration does not outperform Stack_LR_S_Log_WeekSel.

On median MASE, the performance is slightly different. The best method on median MASE is a single base model for all datasets, except the Ausgrid, Solar and Dominick datasets, which also shows the suitability of using a pool of baseline models to forecast weekly series.

However, across the mean sMAPE and mean MASE results, the best performing model on each dataset is the same where across three datasets, Stack_LR_S_Log_WeekSel performs the best. Furthermore, the best performing models across different model groups are also

the same across mean sMAPE and mean MASE results on all datasets except for five cases: M4 dataset best performing models on the first and last groups, Kaggle web traffic dataset best performing model on the first group, Solar dataset best performing model on the third group and Dominick dataset best performing model on the first group. Thus, overall, the best performing models across sMAPE and MASE are very similar, and they only have minor differences.

5.1.4. Reasons for the better performance of Stack_LR_S_Log_WeekSel

Stack_LR_S_Log_WeekSel also shows better performance when compared with the variants shown in Tables A.1 and A.2 in the Appendix that use only the forecasts of the sub-models in WeekSel with different meta-learning algorithms during training. Stack_LinR_S_Log_WeekSel only outperforms Stack_LR_S_Log_WeekSel on Ausgrid and Traffic datasets across all error metrics. Stack_LinR_S_Log_WeekSel also outperforms Stack_LR_S_Log_WeekSel on M4 and NN5 weekly datasets only across median MASE and median sMAPE, respectively. However, overall, Stack_LR_S_Log_WeekSel demonstrates a better performance than Stack_LinR_S_Log_WeekSel on five datasets. Furthermore, Stack_LR_S_Log_WeekSel outperforms Stack_XGB_S_Log_WeekSel on all datasets across all error metrics except for the Solar dataset. Hence, Tables A.1 and A.2 in the Appendix clearly show that in our case, lasso regression is a better option to be used as a meta-learning algorithm to combine sub-model forecasts than other regression techniques such as linear regression and XGBoost. This is because of its capability of sub-model selection (Section 3.3).

Although lasso regression is a robust algorithm that has the capability to select models, when using more suitable sub-models, the final forecasts get more accurate (Section 3.1). Hence, even though the WeekFull model pool contains the four base models in WeekSel, overall, Stack_LR_S_Log_WeekFull provides less accurate forecasts compared with Stack_LR_S_Log_WeekSel. The general model pool has the original FFORMA sub-models. Overall, Stack_LR_S_Log_General provides less accurate forecasts across all datasets than Stack_LR_S_Log_WeekSel and Stack_LR_S_Log_WeekFull showing that including more suitable base models can considerably improve forecast accuracy for weekly series. However, Stack_LR_S_Log_General outperforms FBML_XGB_S_General across all datasets on all error metrics except the mean and median sMAPE of the Traffic dataset and median sMAPE of the Dominick dataset. It further shows that using a lasso regression-based stacking approach is a better option to combine sub-model forecasts in our considered weekly forecasting datasets than the FBML approach (Section 3.2.2). Compared with FBML_XGB_S_General, Stack_LR_S_Log_WeekSel uses more suitable base models and a meta-learning algorithm to produce final forecasts. Hence, Stack_LR_S_Log_WeekSel outperforms FBML_XGB_S_General across all datasets on all error metrics. Furthermore, a linear model such as lasso regression can be more suitable as a forecast combination method than FBML_XGB_S_General, if there is only a small amount of data available for model training (Section 3.3). Stack_LR_S_Log_WeekSel also contains the strengths of both global and local models. As a result, it demonstrates a better performance than FBML_XGB_S_General and the other benchmark models.

FBML_XGB_S_WeekSel has performed better than FBML_XGB_S_General across all experimental datasets on all error metrics except the median sMAPE and mean MASE of the M4 weekly dataset. This further indicates that the base models in WeekSel are more suitable to forecast weekly data than the sub-forecasting models in the General model pool (Section 3.1.3). However, on the NN5, Kaggle web traffic, Traffic and Solar datasets, FBML_XGB_S_WeekSel has not outperformed some base models, underlining the need to explore other combination approaches. Stack_LR_S_Log_WeekSel outperforms FBML_XGB_S_WeekSel across all datasets on all error metrics except for 7 cases: median sMAPE, mean MASE and median MASE of Ausgrid dataset, and all error metrics on the Dominick dataset.

5.1.5. Performance of simple averaging and Stack_LR_S_Log_WeekSel

Generally, averaging the base model forecasts is considered as an efficient and accurate ensembling technique in the forecasting research space. In many cases, simple averaging outperformed FBML_XGB_S_WeekSel, even though the FBML approach starts optimising weights considering the simple average. Major differences between the validation and test sets and small sample sizes can explain this phenomenon. But simple averaging has only outperformed Stack_LR_S_Log_WeekSel on the Ausgrid and Traffic datasets on all error metrics. Averaging does not select the best forecasting models when producing the final forecasts. Furthermore, it assigns equal weights to all base models, and hence, if a subset of models provides poor forecasts, then the final result may considerably be affected by it. This can be a problem, especially if there are models that may be unsuitable, as in our case where we use non-seasonal models to forecast seasonal time series. Stack_LR_S_Log_WeekSel addresses both issues mentioned above of simple averaging, and hence, overall, it has produced more accurate forecasts than the averaging benchmark. The conclusions are the same with other base model pools, where overall, both Average_WeekSel and Stack_LR_S_Log_WeekSel outperform Average_General and Average_WeekFull on all datasets across all error metrics.

5.1.6. Time series features as additional inputs

Next, we analyse whether including time series features as inputs can improve forecasting accuracy. The experiments with features are only conducted with the WeekSel model pool. It is the best base model pool to be used in the weekly forecasting context compared to the other two. The performance of lasso regression models that only take base model forecasts as inputs, namely Stack_LR_S_Log_WeekSel and Stack_LR_PH_Log_WeekSel, is generally better than the performance of lasso regression models that take both forecasts and features as inputs: StackFeat_LR_S_Log_WeekSel, StackFeat_LR_S_

WeekSel, StackFeat_LR_PH_Log_WeekSel and StackFeat_LR_PH_WeekSel. The observations are also the same across the variants that use the other two meta-learning algorithms, linear regression and XGBoost. Hence, including features as inputs in our case does not increase the accuracy. Furthermore, incorporating features to train models can pose additional problems for datasets with short time series as seasonality is not considered during feature calculation.

5.1.7. Number of meta-learners

We also observe that in our case, training a single meta-learner across all horizons provides better results compared to the models that train separate meta-learners per horizon among the variants that use the best model pool, WeekSel, across all three meta-learning algorithms. Furthermore, our experiments show that transforming the base model forecasts with a logarithm is beneficial when training a meta-learning model in a weekly forecasting context.

In summary, Stack_LR_S_Log_WeekSel performs better than all other benchmarks and variants across all datasets. It provides the best forecasts for the M4 weekly dataset with a mean sMAPE of 6.42, which is better than the performance of the winning method of the M4 weekly competition (Darin & Stellwagen, 2020) with a mean sMAPE of 6.58. The M4 weekly winning approach is not an automated forecasting framework. It uses more than 15 sub-model forecasts where domain knowledge and manual inspection are also required to obtain the final forecasts. Compared with the M4 weekly winning approach, Stack_LR_S_Log_WeekSel uses only four sub-models with a fully-automated forecast combination approach without using any specific domain knowledge. Hence, we propose Stack_LR_S_Log_WeekSel as an accurate and a fully-automated strong ensemble model in the weekly time series forecasting space.

5.2. Statistical testing results

Table 4 shows the results of the statistical testing evaluation, namely the adjusted p-values calculated from the Friedman test with Hochberg's posthoc procedure considering a significance level of α=0.05 (García et al., 2010). The overall p-value of the Friedman rank sum test is less than 10^{-30}, which is highly significant. Stack_LR_S_Log_WeekSel performs the best on ranking over sMAPE per series of all experimental datasets, and hence, it is used as the control method as mentioned in the first row. A horizontal line is used to separate the models that perform significantly worse than Stack_LR_S_Log_WeekSel. All individual models, ensemble benchmarks and variants are significantly worse than the control method except Stack_LinR_S_Log_WeekSel as they report p_{Hoch} values less than α.

Table 5 shows a comparison of our proposed method, Stack_LR_S_Log_WeekSel with the M4 weekly winning method, and all other considered benchmarks and variants using the M4 weekly dataset. For each method, we calculate across how many series Stack_LR_S_Log_WeekSel wins, loses, and ties in terms of the sMAPE of each series in the M4 weekly dataset. As shown in Table 5, our proposed method obtains more wins than losses over all other methods, including the winning method of the M4 weekly competition. This further shows that our proposed method is a more accurate weekly forecasting model than all other considered benchmarks and variants.

Table 4
Results of statistical testing.

Model	p_{Hoch}
Stack_LR_S_Log_WeekSel	–
Stack_LinR_S_Log_WeekSel	0.42
Stack_LR_PH_Log_WeekSel	0.01
Stack_LinR_PH_Log_WeekSel	1.74×10^{-3}
Average_WeekSel	1.26×10^{-6}
StackFeat_LR_S_WeekSel	1.22×10^{-6}
Stack_LR_S_Log_WeekFull	7.48×10^{-7}
Stack_LinR_S_Log_WeekFull	4.51×10^{-8}
StackFeat_LR_S_Log_WeekSel	3.95×10^{-9}
StackFeat_LR_PH_Log_WeekSel	5.98×10^{-16}
RNN	$<10^{-30}$
StackFeat_LinR_S_Log_WeekSel	$<10^{-30}$
Stack_LinR_S_Log_General	$<10^{-30}$
FBML_XGB_S_WeekSel	$<10^{-30}$
Average_WeekFull	$<10^{-30}$
Stack_LR_S_Log_General	$<10^{-30}$
Average_General	$<10^{-30}$
TBATS	$<10^{-30}$
ETS	$<10^{-30}$
StackFeat_LinR_PH_Log_WeekSel	$<10^{-30}$
StackFeat_LinR_S_WeekSel	$<10^{-30}$
FBML_XGB_S_General	$<10^{-30}$
Theta	$<10^{-30}$
Auto.ARIMA	$<10^{-30}$
StackFeat_LR_PH_WeekSel	$<10^{-30}$
DHR-ARIMA	$<10^{-30}$
StackFeat_XGB_S_Log_WeekSel	$<10^{-30}$
NNET-AR	$<10^{-30}$
Naïve	$<10^{-30}$
StackFeat_XGB_S_WeekSel	$<10^{-30}$
StackFeat_LinR_PH_WeekSel	$<10^{-30}$
Random Walk	$<10^{-30}$
Stack_XGB_S_Log_WeekFull	$<10^{-30}$
Stack_XGB_S_Log_WeekSel	$<10^{-30}$
StackFeat_XGB_PH_WeekSel	$<10^{-30}$
Stack_XGB_S_Log_General	$<10^{-30}$
Stack_XGB_PH_Log_WeekSel	$<10^{-30}$
StackFeat_XGB_PH_Log_WeekSel	$<10^{-30}$
STLM-AR	$<10^{-30}$
Snaïve	$<10^{-30}$

5.3. Computational performance

We have executed our experiments on different hardware due to their extensiveness, and thus computational times are generally not comparable. To perform a comparison study of computational performance, we execute a subset of the experiments in a controlled environment, namely an Intel(R) Core(TM) i7 processor (2.6 GHz) and 32 GB of main memory.

Table 6 shows the computational times of the considered eleven baseline models and the model that we propose as a strong ensemble weekly forecasting model, Stack_LR_S_Log_WeekSel, which is the best method among

Table 5
Comparison of Stack_LR_S_Log_WeekSel with M4 weekly winning method, and other benchmarks and variants across all series of M4 weekly dataset in terms of the number of times the proposed method wins, loses and ties with the respective comparison method on sMAPE.

Model	Win	Loss	Tie
M4 Weekly Winning Method	195	164	0
Stack_LinR_S_Log_WeekSel	196	162	1
StackFeat_LR_S_Log_WeekSel	199	158	2
Stack_LR_S_Log_WeekFull	206	152	1
FBML_XGB_S_General	206	149	4
Average_General	211	143	5
Stack_LinR_S_Log_WeekFull	212	144	3
Average_WeekSel	213	139	7
TBATS	215	142	2
RNN	215	142	2
FBML_XGB_S_WeekSel	215	144	0
StackFeat_LR_S_WeekSel	216	141	2
ETS	219	139	1
Stack_LR_S_Log_General	219	140	0
StackFeat_LinR_S_Log_WeekSel	222	137	0
Average_WeekFull	223	136	0
Auto.ARIMA	224	134	1
StackFeat_LR_PH_Log_WeekSel	226	133	0
StackFeat_LinR_S_WeekSel	226	132	1
DHR-ARIMA	230	128	1
NNET-AR	230	128	1
Stack_LinR_PH_Log_WeekSel	238	120	1
Stack_LinR_S_Log_General	239	120	0
Naïve	242	117	0
StackFeat_LinR_PH_Log_WeekSel	242	116	1
Random Walk	243	113	3
Stack_LR_PH_Log_WeekSel	243	116	0
Theta	245	113	1
StackFeat_LR_PH_WeekSel	250	107	2
StackFeat_LinR_PH_WeekSel	251	108	0
Snaïve	253	103	3
StackFeat_XGB_S_WeekSel	262	97	0
Stack_XGB_S_Log_General	265	93	1
STLM-AR	268	91	0
StackFeat_XGB_S_Log_WeekSel	273	86	0
Stack_XGB_S_Log_WeekFull	274	84	1
Stack_XGB_S_Log_WeekSel	286	73	0
Stack_XGB_PH_Log_WeekSel	297	62	0
StackFeat_XGB_PH_WeekSel	308	51	0
StackFeat_XGB_PH_Log_WeekSel	317	41	1

all variants across all datasets. The reported computational times of Stack_LR_S_Log_WeekSel include both submodel forecast calculation time and forecast combination time.

From Table 6, we can see that unsurprisingly, comparatively less complex models such as naïve and snaïve show the lowest computation times. The three highly accurate base models in the WeekSel pool, namely TBATS, DHR-ARIMA, and RNN, show considerably higher computational costs than the fourth method, Theta, which executes quickly. Out of all base models, RNN usually shows a much larger execution time except for the M4 and Dominick datasets where DHR-ARIMA and Auto.ARIMA have the highest execution times, respectively. The proposed combination method, Stack_LR_S_Log_WeekSel, requires executing the four base models before computing the final forecasts and therefore shows the highest computational time. However, the overhead over the sum of the base model forecasting times is small.

5.4. Individual model contributions in optimal oracle combination approach

We further conduct a study to explore the contributions of the individual base models to forecasting accuracy. We only consider our best model pool, WeekSel, for this analysis. For this, we assume an optimal oracle combination approach, where the base model with the lowest error (sMAPE or MASE) on the test set is identified for each series and selected for forecasting. This procedure is followed considering the four base models in WeekSel, to identify a lower bound of errors. Then, the same procedure is followed by subsequently removing one of the base models and only using the three remaining ones. This way, we are able to identify base models that perform better than the others on certain series and are not consistently dominated by other methods. Therefore, these models add to the diversity of forecasting methods. Table 7 shows the results of the study. The lower bound results containing the four base models, which is the best-performing method in this comparison, are highlighted in boldface, and the results of the worst-performing combinations are italicised. The combination containing only local models, TBATS_DHR-ARIMA_Theta, is the worst-performing combination for M4, NN5, Kaggle web traffic, Ausgrid and Dominick datasets, and the second-worst combination for the Traffic dataset across all error metrics. The combinations with a mixture of local and global models show better performance than TBATS_DHR-ARIMA_Theta. Thus, we can conclude that the RNN adds diversity to the forecasting pool of methods. Using a mixture of local and global models provides better forecasts as a result of incorporating the strengths of both global and local models while mitigating the weaknesses of each other.

5.5. Analysis of combination weights

Table 8 shows the base model weights chosen by our best stacking-based forecast combination approach, Stack_LR_S_Log_WeekSel across all datasets. TBATS is the best performing base model for M4, and RNN is the best performing base model for the NN5, Kaggle web traffic and Dominick datasets. Stack_LR_S_Log_WeekSel has also assigned the highest weights for the corresponding base models with those datasets and it shows this model has the capability of identifying the best base models that should be used for combining. Furthermore, the method also fully discards some of the base models, when computing the forecasts for Kaggle web traffic, Ausgrid, Traffic, Solar and Dominick datasets. Combining only the best base models while discarding the less important ones is a major benefit of using lasso regression for combining forecasts.

5.6. Applicability of the proposed method to series with other frequencies

Though developed for weekly time series, our methodology applies more broadly to other types of series. For an illustration, we analyse the performance of Stack_LR_

Table 6
Computational times (in seconds) of the baseline models and Stack_LR_S_Log_WeekSel across all datasets. The computational times of Stack_LR_S_Log_WeekSel include both sub-model forecast calculation time and forecast combination time.

	M4	NN5	Kaggle	Ausgrid	Traffic	Solar	Dominick
Theta	3	0.3	2	1	2	0.3	0.2
TBATS	2904	235	1407	516	433	40	28
DHR-ARIMA	11828	165	1910	1987	3839	106	14
RNN	9025	5400	7200	7036	8986	2700	4980
ETS	13	1	9	3	8	1	2
Naïve	1	0.2	2	1	2	0.2	0.05
Snaïve	1	0.2	2	1	2	0.2	0.06
Random Walk	1	0.3	3	1	2	0.4	0.09
STLM-AR	6	1	7	2	44	5	0.2
NNET-AR	1233	9	74	34	36	3	29
Auto.ARIMA	625	1002	1124	1503	322	36	41690
Stack_LR_S_Log_WeekSel	23880	5921	10639	9660	13380	2876	5030

Table 7
Results of the optimal oracle combination study. The results of the lower bound method are highlighted in boldface. The results of the worst performing combinations are italicised.

	M4	NN5	Kaggle	Ausgrid	Traffic	Solar	Dominick
Mean sMAPE							
TBATS_DHR-ARIMA_Theta	*5.99*	*9.95*	*25.84*	*19.63*	10.07	18.41	*14.49*
TBATS_Theta_RNN	5.69	9.40	24.61	19.59	*11.39*	18.89	13.25
Theta_RNN_DHR-ARIMA	5.83	9.37	24.45	18.54	9.93	*20.17*	13.21
TBATS_DHR-ARIMA_RNN	5.71	9.37	24.79	18.39	9.93	18.43	13.18
TBATS_DHR-ARIMA_Theta_RNN	**5.47**	**9.18**	**24.06**	**17.84**	**9.83**	**18.41**	**13.02**
Median sMAPE							
TBATS_DHR-ARIMA_Theta	*3.58*	*9.66*	*23.27*	*14.85*	7.05	17.78	*12.69*
TBATS_Theta_RNN	3.41	8.96	22.49	14.33	*8.74*	18.02	**11.39**
Theta_RNN_DHR-ARIMA	3.41	8.75	22.36	14.04	7.02	*19.60*	11.76
TBATS_DHR-ARIMA_RNN	3.42	8.84	22.68	13.95	7.05	17.78	**11.39**
TBATS_DHR-ARIMA_Theta_RNN	**3.24**	**8.66**	**22.11**	**13.66**	**6.99**	**17.78**	**11.39**
Mean MASE							
TBATS_DHR-ARIMA_Theta	*1.736*	*0.745*	*0.587*	*0.805*	0.853	0.884	*0.811*
TBATS_Theta_RNN	1.664	0.700	0.564	0.800	*1.020*	0.908	0.731
Theta_RNN_DHR-ARIMA	1.648	0.699	0.562	0.758	0.843	*0.978*	0.737
TBATS_DHR-ARIMA_RNN	1.648	0.699	0.571	0.761	0.846	0.886	0.733
TBATS_DHR-ARIMA_Theta_RNN	**1.561**	**0.684**	**0.555**	**0.732**	**0.837**	**0.884**	**0.724**
Median MASE							
TBATS_DHR-ARIMA_Theta	*1.170*	*0.708*	*0.465*	*0.646*	0.676	0.874	*0.723*
TBATS_Theta_RNN	1.097	0.636	0.447	0.606	*0.855*	0.892	**0.621**
Theta_RNN_DHR-ARIMA	1.113	0.655	0.446	0.618	0.666	*0.957*	0.650
TBATS_DHR-ARIMA_RNN	1.059	0.655	0.448	0.601	0.669	0.874	0.674
TBATS_DHR-ARIMA_Theta_RNN	**1.043**	**0.620**	**0.437**	**0.571**	**0.664**	**0.874**	**0.621**

Table 8
Base Model Weights Chosen by Stack_LR_S_Log_WeekSel.

	M4	NN5	Kaggle	Ausgrid	Traffic	Solar	Dominick
TBATS	0.712	0.408	0.006	0.219	0.226	0.042	0.0
DHR-ARIMA	0.009	0.042	0.0	0.0	0.110	0.003	0.0
Theta	0.044	0.049	0.0	0.0	0.0	0.950	0.321
RNN	0.238	0.477	0.976	0.642	0.633	0.0	0.676

S_Log_WeekSel across datasets of other frequencies such as daily data. Table 9 shows the performance of that model and its variants that use the same base model pool and meta-learning algorithm across two daily datasets, namely the NN5 daily dataset and the Kaggle web traffic daily dataset, where both datasets are the original daily versions of the weekly datasets we considered before aggregation. The best-performing models are highlighted in boldface.

Our proposed model, Stack_LR_S_Log_WeekSel, shows the best performance on the Kaggle web traffic daily dataset compared with the performance of sub-models and other variants across all error metrics. On the NN5 daily dataset, another variant of the model, StackFeat_LR_S_WeekSel, shows the best performance across all error metrics except median sMAPE. Hence, the general methodology is applicable and may serve as a good ensemble model more broadly and for data of other frequencies.

Table 9
Results of our best model, Stack_LR_S_Log_WeekSel and its variants across NN5 and Kaggle web traffic daily datasets. The best performing models of each dataset on each error metric are highlighted in boldface.

	Mean sMAPE	Median sMAPE	Mean MASE	Median MASE
NN5 Daily				
ETS	21.57	20.35	0.860	0.810
Theta	21.93	20.51	0.885	0.838
TBATS	21.11	**19.56**	0.858	0.834
DHR-ARIMA	26.87	24.65	1.078	1.055
RNN	22.03	20.65	0.867	0.825
StackFeat_LR_PH_WeekSel	22.39	21.86	0.905	0.879
StackFeat_LR_PH_Log_WeekSel	23.83	22.36	0.962	0.943
Stack_LR_PH_Log_WeekSel	22.78	22.14	0.912	0.909
StackFeat_LR_S_WeekSel	**21.00**	20.21	**0.828**	**0.804**
StackFeat_LR_S_Log_WeekSel	21.57	20.84	0.860	0.845
Stack_LR_S_Log_WeekSel	21.47	19.76	0.854	0.835
Kaggle Daily				
ETS	53.50	47.63	1.580	0.910
Theta	43.13	40.23	0.957	0.758
TBATS	42.52	39.42	0.848	0.730
DHR-ARIMA	44.03	40.86	0.952	0.758
RNN	39.38	37.55	0.788	0.695
StackFeat_LR_PH_WeekSel	51.11	47.67	1.330	0.907
StackFeat_LR_PH_Log_WeekSel	40.69	39.06	0.821	0.722
Stack_LR_PH_Log_WeekSel	40.70	38.96	0.818	0.725
StackFeat_LR_S_WeekSel	44.58	41.61	0.952	0.776
StackFeat_LR_S_Log_WeekSel	**39.30**	37.59	0.788	**0.694**
Stack_LR_S_Log_WeekSel	**39.30**	**37.50**	**0.787**	**0.694**

6. Conclusions and future research

Nowadays, many businesses and industries deal with weekly data and need accurate forecasts on those. However, the forecasting literature currently lacks easy-to-use, accurate, and automated approaches dedicated to forecasting weekly series. In this paper, we have analysed to identify a strong automated ensemble model suitable to forecast weekly time series based on the most recent trends in forecasting. We have analysed two meta-learning architectures, FBML and stacking, with three meta-learning algorithms: linear regression, lasso regression, and XGBoost, as well as three different base model pools. Based on all considered model variants, the stacking approach that optimally combines the forecasts of four base models: RNN, Theta, TBATS and DHR-ARIMA using lasso regression provided the best forecasts overall across seven experimental datasets.

Our best performing model has also significantly outperformed a series of benchmarks such as simple averaging, FFORMA and the current state-of-the-art weekly forecasting models: TBATS and DHR-ARIMA. In particular, the suggested lasso regression model ranks first among the original contenders of the M4 weekly competition, based on mean sMAPE. Furthermore, it applies to any weekly dataset irrespective of the series length. It does not require time series features during model training where, to calculate features with proper seasonal handling, the series length should be at least two seasonal cycles. Hence, we conclude that our weekly forecasting model can be used as an easy-to-implement and automatic, strong ensemble model in the weekly time series forecasting space.

The success of this model encourages future work to further analyse the approach with data of other frequencies, where our preliminary experiments have shown that this model can be a possible approach to obtain accurate forecasts for daily data. Furthermore, it is worth exploring the methodological changes required to scale this method to larger datasets. Some possible changes would be replacing TBATS and the RNN with other forecasting models such as LightGBM that scale better with larger datasets and using XGBoost or LightGBM as the meta-learning algorithm instead of lasso regression.

Declaration of competing interest

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

Acknowledgments

This research was supported by the Australian Research Council under grant DE190100045, a Facebook Statistics for Improving Insights and Decisions research award, Monash University Graduate Research funding and the MASSIVE High performance computing facility, Australia.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ijforecast.2022.01.008.

References

Al-qaness, M., Ewees, A., Fan, H., & Abd Elaziz, M. (2020). Optimized forecasting method for weekly influenza confirmed cases. *International Journal Of Environmental Research And Public Health*, *17*(10), 3510.

Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta model: a decomposition approach to forecasting. *International Journal Of Forecasting*, *16*(4), 521–530, The M3- Competition.

AusGrid (2019). Solar home electricity data. URL https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data, Accessed: 2020-05-10.

Bandara, K., Bergmeir, C., & Hewamalage, H. (2019). LSTM-MSNet: leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE Transactions On Neural Networks And Learning Systems*.

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: a clustering approach. *Expert Systems with Applications*, *140*, 112896.

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., & Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-Term memory neural network methodology. In *26th International Conference On Neural Information Processing*(pp. 462–474).

Barakat, E. H., & Al-Qasem, J. M. (1998). Methodology for weekly load forecasting. *IEEE Transactions On Power Systems*, *13*(4), 1548–1555.

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

Box, G., & Jenkins, G. (1990). *Time series analysis, forecasting and control*. HoldenDay Inc..

Caltrans (2020). Caltrans performance measurement system, california department of transportation. URL http://pems.dot.ca.gov, Accessed: 2020-04-30.

Center, J. M. K. (2020). Dominick's dataset. https://www.chicagobooth.edu/research/kilts/datasets/dominicks.

Cerqueira, V., Torgo, L., Oliveira, M., & Pfahringer, B. (2017). Dynamic and heterogeneous ensembles for time series forecasting. In *IEEE International Conference On Data Science And Advanced Analytics*(pp. 242–251).

Chandra, A., Chen, H., & Yao, X. (2006). Trade-off between diversity and accuracy in ensemble generation. In Y. Jin (Ed.), *Multi-objective machine learning* (pp. 429–464). Berlin, Heidelberg: Springer.

Chen, T., & Guestrin, C. (2016). XGBoost: a scalable tree boosting system. In *KDD '16*, *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY, USA: Association for Computing Machinery.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., & Li, Y. (2020). xgboost: extreme gradient boosting. URL https://CRAN.R-project.org/package=xgboost, R package version 1.2.0.1.

Darin, S. G., & Stellwagen, E. (2020). Forecasting the M4 competition weekly data: Forecast Pro's winning approach. *International Journal Of Forecasting*, *36*(1), 135–141.

de Menezes, L. M., Bunn, D. W., & Taylor, J. W. (2000). Review of guidelines for the use of combined forecasts. *European Journal Of Operational Research*, *120*(1), 190–204.

Diebold, F. X., & Shin, M. (2019). Machine learning for regularized survey forecast combination: partially-egalitarian LASSO and its derivatives. *International Journal Of Forecasting*, *35*(4), 1679–1691.

Divina, F., Gilson, A., Gómez-Vela, F., Torres, M. G., & Torres, J. F. (2018). Stacking ensemble learning for short-term electricity consumption forecasting. *Energies*, *11*, 949.

Duncan, G. T., Gorr, W. L., & Szczypula, J. (2001). Forecasting analogous time series. In J. S. Armstrong (Ed.), *vol. 30*, *Principles of forecasting*. Boston, MA: Springer.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Flunkert, V., Salinas, D., & Gasthaus, J. (2017). DeepAR: probabilistic forecasting with autoregressive recurrent networks. *International Journal Of Forecasting*, *36*(3), 1181–1191.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal Of Statistical Software*, *33*(1), 1–22.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Information Sciences*, *180*(10), 2044–2064.

Gers, F., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: continual prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471.

Godahewa, R., Deng, C., Bergmeir, C., & Prouzeau, A. (2020). Simulation and optimisation of air conditioning systems using machine learning. https://arxiv.org/abs/2006.15296.

Google (2017). *Web Traffic Time Series Forecasting*. URL https://www.kaggle.com/c/web-traffic-time-series-forecasting.

Guttormsen, A. G. (1999). Forecasting weekly salmon prices: risk management in fish farming. *Aquaculture Economics & Management*, *3*(2), 159–166.

Harvey, A. C., & Shephard, N. (1993). 10 Structural time series models. *Handbook Of Statistics*, *11*, 261–302.

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: current status and future directions. *International Journal Of Forecasting*.

Hurvich, C. M., & Tsai, C. (1993). A corrected akaike's information criterion for vector autoregressive model selection. *Journal Of Time Series Analysis*, *14*(3), 271–279.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In C. A. C. Coello (Ed.), *Learning And Intelligent Optimization* (pp. 507–523). Berlin, Heidelberg: Springer.

Hyndman, R. J. (2008). *Forecasting With exponential smoothing: The state space approach*. Springer.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd). OTexts.

Hyndman, R. J., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., & Yasmeen, F. (2020). Forecast: forecasting functions for time series and linear models. R package version 8.12. http://pkg.robjhyndman.com/forecast.

Hyndman, R. J., & Billah, B. (2003). Unmasking the theta method. *International Journal Of Forecasting*, *19*(2), 287–290.

Hyndman, R. J., Kang, Y., Talagala, T., Wang, E., & Yang, Y. (2019). Tsfeatures: time series feature extraction. R package version 1.0.1. https://pkg.robjhyndman.com/tsfeatures.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal Of Statistical Software, Articles*, *27*(3), 1–22.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal Of Forecasting*, *22*(4), 679–688.

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal Of Forecasting*, *36*(1), 167–177.

Khairalla, M. A., Ning, X., AL-Jallad, N. T., & El-Faroug, M. O. (2018). Short-term forecasting for energy consumption through stacking heterogeneous ensemble learning model. *Energies*, *11*, 1–21.

Kolassa, S. (2011). Combining exponential smoothing forecasts using akaike weights. *International Journal Of Forecasting*, *27*(2), 238–251.

Lai, G. (2017). Multivariate-time-series-data. In *GitHub*. Accessed: 2020-05-04, https://github.com/laiguokun/multivariate-time-series-data.

Landeras, G., Ortiz-Barredo, A., & López, J. J. (2009). Forecasting weekly evapotranspiration with ARIMA and artificial neural network models. *Journal Of Irrigation And Drainage Engineering*, *135*(3), 323–334.

Livera, A. M. D., Hyndman, R. J., & Snyder, R. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, *106*(496), 1513–1527.

Makridakis, S., & Hibon, M. (2000). The M3-competition: results, conclusions and implications. *International Journal Of Forecasting*, *16*(4), 451–476.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 competition: results, findings, conclusion and way forward. *International Journal Of Forecasting*, *34*(4), 802–808.

Mcompetitions (2018). M4-methods. URL https://github.com/Mcompetitions/M4-methods.

Mohanty, S., Jha, M., Raul, S. K., Panda, R., & Sudheer, K. (2015). Using artificial neural network approach for simultaneous forecasting of weekly groundwater levels at multiple sites. *Water Resources Management*, *29*.

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: feature-based forecast model averaging. *International Journal Of Forecasting*, *36*(1), 86–92.

Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: locality and globality. *International Journal Of Forecasting*.

O'Hara-Wild, M., & Hyndman, R. J. (2018). Fasster. https://fasster.tidyverts.org.

Orabona, F., & Tommasi, T. (2017). Training deep networks without learning rates through coin betting. *Advances In Neural Information Processing Systems*, *30*, 2160–2170.

Oussalah, M., & Zaidi, A. (2018). Forecasting weekly crude oil using Twitter sentiment of U.S. foreign policy and oil companies data. In *2018 IEEE International Conference On Information Reuse And Integration (IRI)* (pp. 201–208).

Padt, F., & Bergmeir, C. (2017). Optical retail clustering assisted hierarchical forecasting. In *Invited Practitioner Talk, International Symposium On Forecasting (ISF) 2017, 25-28/6/2017 Cairns, Australia*.

Pan, B., & Yang, Y. (2017). Forecasting destination weekly hotel occupancy with big data. *Journal Of Travel Research*, *56*(7), 957–970.

Pawlikowski, M., & Chorowska, A. (2020). Weighted ensemble of statistical models. *International Journal Of Forecasting*, *36*(1), 93–97.

Sanchez, I. (2008). Adaptive combination of forecasts with application to wind energy. *International Journal Of Forecasting*, *24*(4), 679–693.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal Of Forecasting*, *36*(1), 75–85.

Smyl, S., & Kuber, K. (2016). Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks. In *36th International Symposium On Forecasting*.

Solar (2020). Solar power data for integration studies, national renewable energy laboratory. URL https://www.nrel.gov/grid/solar-power-data.html, Accessed: 2020-04-30.

Suilin, A. (2017). Kaggle-web-traffic. In *GitHub*. Accessed: 2020-05-15, https://github.com/Arturus/kaggle-web-traffic.

Svetunkov, I. (2020). Smooth: forecasting using state space models. URL https://CRAN.R-project.org/package=smooth, R package version 2.6.0.

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems With Applications*, *39*(8), 7067–7083.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal Of The Royal Statistical Society, Series B*, *58*, 267–288.

Timmermann, A. (2006). Forecast combinations. *Handbook Of Economic Forecasting*, *1*, 135–196.

Trapero, J. R., Kourentzes, N., & Fildes, R. (2015). On the identification of sales forecasting models in the presence of promotions. *Journal Of The Operational Research Society*, *66*(2), 299–307.

Wilms, I., Rombouts, J., & Croux, C. (2016). Lasso-based forecast combinations for forecasting realized variances. *Econometric Modeling: Capital Markets - Forecasting EJournal*.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Zhai, B., & Chen, J. (2018). Development of a stacked ensemble model for forecasting and analyzing daily average PM2.5 concentrations in Beijing, China. *Science Of The Total Environment*, *635*, 644–658.