Ensembles of Localised Models for Time Series Forecasting

Rakshitha Godahewa[a,*], Kasun Bandara[b], Geoffrey I. Webb[a], Slawek Smyl[c], Christoph Bergmeir[a]

[a] *Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Monash University, Melbourne, Australia*
[b] *School of Computing and Information Systems, Melbourne Centre for Data Science, University of Melbourne, Melbourne, Australia.*
[c] *Uber Technologies Inc, San Francisco, California, United States*

Abstract

With large quantities of data typically available nowadays, forecasting models that are trained across sets of time series, known as Global Forecasting Models (GFM), are regularly outperforming traditional univariate forecasting models that work on isolated series. As GFMs usually share the same set of parameters across all time series, they often have the problem of not being localised enough to a particular series, especially in situations where datasets are heterogeneous. We study how ensembling techniques can be used with generic GFMs and univariate models to solve this issue. Our work systematises and compares relevant current approaches, namely clustering series and training separate submodels per cluster, the so-called ensemble of specialists approach, and building heterogeneous ensembles of global and local models. We fill some gaps in the existing GFM localisation approaches, in particular by incorporating varied clustering techniques such as feature-based clustering, distance-based clustering and random clustering, and generalise them to use different underlying GFM model types. We then propose a new methodology of clustered ensembles where we train multiple GFMs on different clusters of series, obtained by changing the number of clusters and cluster seeds. Using Feed-forward Neural Networks, Recurrent Neural Networks, and Pooled Regression models as the underlying GFMs, in our evaluation on eight publicly available datasets, the proposed models are able to achieve significantly higher accuracy than baseline GFM models and univariate forecasting methods.

Keywords: Time Series Forecasting, Feed-Forward Neural Networks, Recurrent Neural Networks, Pooled Regression, Ensemble Models

1. Introduction

Nowadays, many businesses and industries, such as retail, energy, and tourism, routinely collect massive amounts of time series from similar sources that require accurate forecasts for

*Corresponding author. Postal Address: Faculty of Information Technology, P.O. Box 63 Monash University, Victoria 3800, Australia. E-mail address: rakshitha.godahewa@monash.edu

better decision making and strategic planning. Recently, global forecasting models (GFM) that build a single model across many time series [1], have shown their potential in providing accurate forecasts under these circumstances, e.g., by winning the prestigious M4 and M5 forecasting competitions [2, 3]. Compared with traditional univariate forecasting methods, such as Exponential Smoothing [ETS, 4] and Auto-Regressive Integrated Moving Average models [ARIMA, 5] that build separate models to forecast each series, GFMs are built using all the available time series, and are capable of exploiting cross-series information and control overall model complexity better through shared parameters [6], though their complexity is often required to be controlled based on heterogeneity and amount of the training data available [7].

A GFM in its purest form, e.g., a pooled regression model [PR, 8, 9], shares all parameters across all series, and can therewith be seen as an extreme opposite case of a traditional univariate forecasting model that has only per-series parameters. Such GFMs now consequently often have the problem that they are not localised enough for particular series, especially when the time series datasets are heterogeneous. To address this issue, researchers have developed special models with both global parameters shared across all series and parameters per each series [10, 11]. However, another more generally applicable and generic approach to address this problem that can be used with any GFM is to build models across subsets of the overall time series database. This strategy allows both to control model complexity and to build more localised models, focused on certain groups of time series. The challenge then is how to define these subgroups of series and how to combine the outputs of the resulting submodels.

Clustering is one approach that can be used, both as a way to make the overall model more complex in a controlled way and to address data heterogeneity. Montero-Manso and Hyndman [6] show that random clustering can improve forecasting accuracy, as it increases overall model complexity. To address data heterogeneity, Bandara et al. [12] propose an approach that initially clusters the time series based on a similarity measure, and then trains separate GFMs on each identified cluster. Similarly, clusters can be defined using domain knowledge, for example by using spatial proximity in applications such as environmental prediction and weather prediction [13]. The winning method of the M4 forecasting competition [10] uses an ensembling approach known as *ensemble of specialists*. This approach trains a group of GFMs that specialise by being more likely to train on series they already perform better on than the other models in the ensemble. Yet another approach is to combine global and local models in a forecast combination approach, as in [14].

The first main contribution of our study is to systematically compare the different generic approaches of localising GFMs proposed in the literature, namely clustering, ensemble of specialists, and forecast combination of local and global models. Therefore, we generalise the ensemble of specialists and clustering approaches from the literature, using them with different baseline GFM architectures. Also, the current work in the literature only uses random and feature-based clustering approaches for subgrouping the time series. We fill this gap by additionally using distance-based clustering.

In machine learning, ensembling techniques are used very successfully to reduce model variance and model bias, and to quantify the uncertainty by aggregating predictions over

multiple models [see, e.g., 15, 16]. But also in the field of forecasting, it is well established for decades that forecast combinations lead to improved accuracy [17, 18]. Forecast combinations address the limitations following from the *No Free Lunch Theorem* by Wolpert [19] which states that there is no single best prediction algorithm that is suitable for any application. Forecast combinations containing diverse models [20] incorporate the strengths of different models and hence, they are capable of providing accurate forecasts for many applications compared with single forecasting models.

This motivates us to propose an ensembling approach to localise GFMs which is the second main contribution of this study. Our proposed ensemble models contain localised GFMs trained over subsets of collections of time series with different model complexities. In particular, we first subgroup the time series using two feature-based clustering approaches: K-means [21] and K-means++ [22], and a distance-based clustering approach, K-medoids [23] with Dynamic Time Warping (DTW) distance in multiple iterations by either varying the number of clusters or the cluster seed. Then, multiple GFMs are trained per each identified subgroup, for multiple iterations, generating multiple forecasts for each time series. These forecasts are then averaged to obtain the final predictions. Here, the issue of information loss in traditional clustering-based GFM approaches, as series are assigned to either one cluster or another and only one model is built per cluster, is addressed by building multiple GFMs over different clusters. We also show that more sophisticated clustering is a good way to add model complexity in a directed and focused way, especially if the amount of data is limited. Furthermore, we quantify in our extensive experimental study how localisation and increasing model complexity affect the performance of GFMs. As the primary forecasting modules of our framework, we use linear and non-linear GFM models, namely: PR models, Feed-Forward Neural Networks (FFNN), and Recurrent Neural Networks (RNN). All implementations of this study are publicly available at: `https://github.com/rakshitha123/Localised_Ensembles`.

The remainder of this paper is organized as follows: Section 2 reviews the related work. Section 3 introduces the problem statement and the methods used for the experiments such as FFNNs, RNNs with Long Short-Term Memory (LSTM) cells, PR models, and ensemble and non-ensemble models. Section 4 explains the experimental framework, including the datasets, hyperparameter tuning, error metrics and benchmarks. We present an analysis of the results in Section 5. Section 6 concludes the paper and discusses possible future research.

2. Related Work

In the following, we discuss related work and the theoretical basis in the areas of global models, localisation of global models, ensembling, and time series clustering.

2.1. Global Models in Forecasting

GFMs [1] are a relatively recent trend in forecasting and have become popular after being the base of the winning solution of the M4 forecasting competition [10]. Usually, GFMs build a single model with a set of global parameters across many series. In contrast to local models, they are capable of learning the cross-series information during model training with

a smaller amount of parameters. Montero-Manso and Hyndman [6] show that the complexity of local models increases with the number of series as a separate model is fitted for each series whereas the complexity of global models remains the same regardless of the amount of series. Those authors also prove that for a particular set of series, there always exists a global model that can produce the same forecasts as given by a collection of local models. Thus, in principle GFMs are applicable to any forecasting problem, not only in situations where series are related, as in earlier work typically postulated [7]. However, the proof does not provide a way to construct such global models, so that in practice, there are still complex interdependencies between the degree of relatedness of series, the amount of data available, and the complexity of the models [7]. Hewamalage et al. [7] perform a simulation study to characterise these interdependencies, and define relatedness using a model-based definition, in the way that series are defined to be related if their data generating processes are similar.

In terms of particular GFMs proposed in the literature, in an early work, Duncan et al. [24] use a Bayesian pooling approach to extract information from similar time series to train GFMs. A linear auto-regressive GFM trained across a set of time series is known as a PR model, as discussed by Trapero et al. [8]. Smyl and Kuber [25] use global RNNs for forecasting. Salinas et al. [26] propose a global auto-regressive RNN named DeepAR for probabilistic forecasting. Since 2018, many more models have been proposed in this space, e.g., Oreshkin et al. [27] propose the forecasting framework N-BEATS which trains deep neural networks incorporating cross-series information, and Hewamalage et al. [28] present an overview of globally trained RNNs. The M5 forecasting competition [3] was won by a globally trained gradient-boosted tree, and also all recent Kaggle competitions in forecasting were won by global models [29].

Though GFMs may be relatively new in their appreciation in forecasting, similar concepts have been used in other fields, most prominently pooled regression models [9], e.g., in the social sciences, and sub-population models in medicine [30]. Gelman and Hill [9] argue that developing prediction models with pooling and without pooling data are extreme opposite cases where both approaches have their own issues. The no-pooling approach tends to overfit the data, especially for instances with a smaller amount of training data. On the other hand, pooling or a global approach may underfit the data and tends to provide averaged predictions for all instances and therewith it will ignore specifics of individual instances (see also Kent and Hayward [30]). Thus, those authors argue that the middle ground between pooling and no-pooling approaches is often more desirable, which in their context is called *multilevel analysis*. In line with this concept, in our work, we use a middle ground between global (pooling) and local (no-pooling) approaches where we train global models across sets of time series, ensuring that the models are adequately localised by using more appropriate time series groups to train them.

2.2. Localisation of Global Models

Some recent works in the literature introduce models with both global and local parameters, with the most relevant works in this space being DeepGLO [11] and the winning approach of the M4 forecasting competition, Exponential Smoothing-Recurrent Neural Network [ES-RNN, 10]. However, we focus in our work on methods that can be generically

applied to existing GFM models, to enable them to achieve more localised learning.

ES-RNN not only has local and global parameters, but it also has a mechanism for localisation, called the ensemble of specialists. This architecture first assigns the series randomly to train a set of GFM experts. Then, it identifies the most suitable series to train each specialist based on the errors corresponding to a validation dataset. This procedure is iteratively repeated until the average forecasting error on the validation set starts growing. Finally, forecasts for each time series are computed by aggregating the forecasts of GFM specialists that correspond to each time series. In this approach, data heterogeneity is addressed by selecting the most suitable series to train each specialist, and generating the final forecasts using multiple GFM specialists.

The only prior work that uses time series clustering for localising and addressing data heterogeneity of GFMs, to the best of our knowledge, is the work of Bandara et al. [12]. This work implements feature-based clustering approaches using K-means [21], DBScan [31], Partition Around Medoids [PAM, 32] and Snob [33] as base clustering algorithms. For each identified cluster, those authors build a separate GFM, using RNN as the base forecasting algorithm. That work also has been extended to address real-world forecasting challenges in retail and healthcare [34, 35]. However, these approaches have the limitation of information loss that may occur due to the suboptimal grouping of time series. As the method makes a hard binary decision about which GFM to use for a particular time series, based on the clustering, a wrong selection here leads to a sub-optimal model being used. Our proposed framework addresses this limitation by aggregating the forecasts generated by a set of clusters in multiple iterations. On top of this, we use both feature-based and distance-based clustering techniques to identify subgroups of time series, where prior work has been only based on feature-based clustering.

Montero-Manso and Hyndman [6] see clustering as a method to add more complexity to a GFM. Those authors show that increasing model complexity can improve the forecasting accuracy, thus localising GFMs by using techniques such as clustering supports to improve the performance of GFMs. They only consider random clustering for their experiments. We use their work as a theoretical base and a starting point, also addressing data heterogeneity by using more sophisticated clustering techniques, and therewith also increasing model complexity in a directed and focused way.

The main notion of GFMs is typically that all series used for training are relevant for evaluation, which is why it is beneficial to develop a GFM that will perform well on average across these series (and localise accordingly). Another related topic is data augmentation, where, e.g., Bandara et al. [36] use simulated series to improve the performance of GFMs. However, as here the GFM does not need to perform well on the augmented series, here the GFM needs to be trained in a pooled way across real and augmented data, without localisation, so that the GFM can generalise across both real and augmented data, to prevent overfitting on the real data.

2.3. Ensembling for Forecasting: Forecast Combinations

In time series forecasting, ensembling is known under the name of forecast combination [18, 37, 38]. As in many ensembling techniques, forecast combination techniques usually

train multiple submodels independently, and then aggregate over the submodels to compute final forecasts. The submodel aggregation technique plays a crucial role in forecast combination approaches. Simple averaging and weighted averaging are widely used to aggregate forecasts [18].

Krogh and Vedelsby [39] show that the squared error of a linearly combined ensemble model can be explained as the addition of two distinct components: average squared errors of the individual base models and a term that quantifies the interactions or diversity among the base model predictions. Those authors show that this second term which represents the diversity is always positive. Thus, the squared error of the ensemble model is always less than or equal to the average squared error of the individual base models. This phenomenon shows that higher diversity of base models produces a larger gain for the ensemble model over the average performance of the individual base models. Furthermore, Brown et al. [40] show that diversity acts as an extra degree of freedom in the bias-variance trade-off which allows the ensemble model to find functions that are hard to find with an individual forecasting model. Thus, ensemble models are expected to provide better results when submodel forecasts are uncorrelated, and heterogeneous base learners are more likely to generate uncorrelated forecasts [41, 42]. Torgo and Oliveira [43] propose an ensemble of bagged trees approach, where diversity among the model trees is increased by using different time series representations. Cerqueira et al. [41] propose an ensemble approach with dynamic heterogeneous submodels that uses a weighting scheme. These heterogeneous ensemble models have been used to address many real-world forecasting problems, such as power forecasting [44], electricity load forecasting [45], and price forecasting [46].

Ensembling and localisation have been used together to improve the prediction accuracy. Masoudnia and Ebrahimpour [47] discuss prediction methods that incorporate a mixture of experts concept where multiple neural networks are trained for different parts of the problem space. The work by Laurinec et al. [48] provides insights on how clustering based ensembling techniques can be used specifically to forecast electricity consumption.

Meta-learning based weighted ensembles are also popular. The second-winning approach in the M4 forecasting competition, Feature-based Forecast Model Averaging [FFORMA, 49] uses a weighted ensembling approach that combines base models using a meta-learner, which is trained using time series features. The third-winning approach [50] also uses an ensemble technique to optimally combine the forecasts generated by a set of statistical models. The meta-learning approach proposed by Cerqueira et al. [51] uses the prediction of the loss of sub-models to determine the weights that should be used when combining their forecasts.

2.4. Time Series Clustering

Clustering, or identifying similar groups of data that have similar characteristics, is a widely used approach to handle data heterogeneity [52]. Three main classes of time series clustering approaches are distance-based, feature-based, and model-based approaches [53]. The distance-based clustering techniques group time series based on the similarity of raw observations of time series. Here, the similarity of point values is captured using a distance metric, such as Euclidean distance, Manhattan distance, Pearson correlation, Spearman correlation, or DTW. As the performance of distance-based clustering techniques depends

heavily on the distance metric used, choosing the appropriate distance measure can be a challenging task in this approach. In contrast, feature-based clustering techniques first extract a set of features from each time series. Then, a traditional clustering technique is applied to the extracted feature matrix to identify similar groups of time series. Compared to distance-based clustering, feature-based clustering approaches are typically more interpretable and more resilient to missing data [54]. On the other hand, model-based clustering assumes the data are originally generated from a group of models and therefore attempts to recover the models from the data [55]. Here, expectation-maximisation algorithms are used to assign each series to the cluster with maximum likelihood.

3. Methods

This section first gives a brief overview of the proposed methodology and the primary prediction models used to implement the GFMs in this study and their corresponding preprocessing techniques. Then, we describe the GFM-based ensemble variants introduced.

3.1. Methodology Overview: Localisation of GFMs Through Ensembling

Time series forecasting refers to predicting the future values of a given set of time series based on their past values. In our experimental framework, we consider a dataset as a set of time series. As shown in Figure 1, we split each time series in a dataset into training and test parts where the training parts are used to train GFMs and the test parts represent the actual values corresponding with the expected forecast horizon. In the GFM context, the training set consists of the past values from a group of time series, whereas the test set consists of their corresponding future values.

Our proposed framework first clusters a given collection of time series based on their similarity, and then trains a set of localised GFMs, one per each time series cluster, where the training set of each GFM consists of the training parts of the corresponding time series cluster. The future values of each time series are obtained using their corresponding localised GFM. This procedure is repeated multiple times either by changing the number of clusters or cluster seed. Therefore, for each series, we obtain multiple forecasts provided by a set of localised GFMs in multiple iterations. The final forecasts for a given series are obtained by averaging the forecasts provided by the corresponding localised GFMs. Though there is research in the literature around the use of weighted averages and other techniques to obtain the final forecasts [38, 41], and there are situations where weighted averaging provides superior results over simple averaging [49, 14], a consensus in the forecasting literature is that a simple average is hard to beat and often performs best [18, 56]. Thus, we limit our experiments to simple averages as we deem this particular question not to be the main focus of our paper.

3.2. Global Forecasting Models

We use FFNNs, RNNs, and PR models as the baseline GFMs in our work. FFNNs have the universal function approximation property and therewith are capable of approximating any (non-linear) function [57]. RNNs belong to the NN family and they are especially

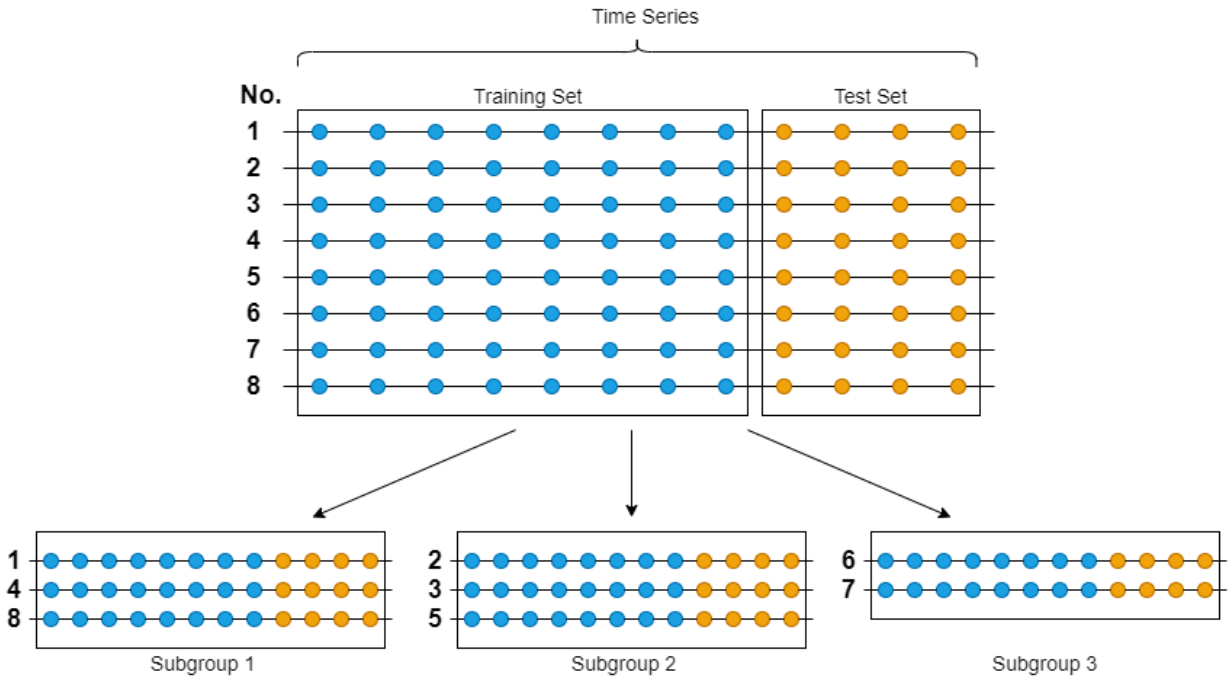

Figure 1: Visualisation of the training and test sets in GFM context. Series are clustered into different groups based on their similarity and a localised GFM is trained per each series cluster.

suitable for sequence modelling problems [58] due to their capability of addressing temporal order and temporal dependencies of sequences. Furthermore, global RNNs have been recently used as a central part of the winning method of the M4 forecasting competition, making them one of the state-of-the-art models in the forecasting space [10]. As both FFNNs and RNNs are non-linear GFMs, we also consider linear GFMs in our work. Hence, we also consider PR models [8] in our work which use linear functions of lagged values of time series to predict their future. Further details of these GFMs are detailed in the following.

3.2.1. Feed-Forward Neural Networks

FFNN is a strong non-linear standard model, which has the universal function approximation property [57]. For further details of FFNNs, we refer to Goodfellow et al. [59]. In our work we use the implementation available in R in the `nnet` function from the `nnet` package [60]. To use an FFNN for time series forecasting, the time series are preprocessed as explained in Section 3.2.4. We then train the FFNNs using the single-step ahead forecasting strategy, where models are iteratively used to obtain the forecasts for the required forecast horizon [61].

3.2.2. Recurrent Neural Networks with Long Short-Term Memory Cells

RNNs are a type of NNs that are especially suited for sequence modelling [58]. In RNNs, feedback loops address the temporal order and temporal dependencies of sequences, which may be less aptly handled by traditional NNs [62]. Recently, RNN-based GFMs are

becoming a popular NN architecture among forecasting practitioners [10, 12, 28, 34, 63, 26], and have been used as the primary architecture in the winning solution of the M4 forecasting competition. Based on recommendations by Hewamalage et al. [28], we use the LSTM network as our primary RNN architecture and implement the Stacking Layers design pattern to train the network [64, 14]. However, we highlight that the proposed methodology can be generalised to any RNN variant such as Elman RNN [58], Gated Recurrent Units [GRU, 65], and others. We implement the proposed RNN architectures using `TensorFlow` [66], based on the framework developed by Hewamalage et al. [28]. Similar to FFNN, time series are preprocessed as discussed in Section 3.2.4. Then, we use the RNN stacked architecture with the multi-step ahead forecasting strategy to train the network, following the procedure recommended by Hewamalage et al. [28] .

3.2.3. Pooled Regression Models

PR models [8] are, similar to FFNNs and in contrast to RNNs, purely auto-regressive, and use a linear function of the lagged values of a time series to predict its future. They estimate a set of constant coefficients corresponding to each considered lagged value. PR models train on a pool of time series during model training and the coefficients are computed by minimising the overall training error. The lagged values are then multiplied by their corresponding coefficients and an intercept is added to obtain the final forecasts. PR models use the single-step ahead forecasting strategy and therefore the models are iteratively used to obtain the forecasts for the required forecast horizon. In our experiments, we use the `glm` function in the `glmnet` package [67] to implement the proposed PR variants.

3.2.4. Preprocessing for Global Forecasting Models

For FFNN and RNN models, the time series are first normalised by dividing them by their corresponding mean values. Next, the normalised series are transformed to a log scale to stabilise the variance and to avoid placing the outputs in the saturated areas of the NN's activation function. There are two main approaches to handle the seasonality of time series when using RNNs: deseasonalisation or using original observations with seasonal exogenous variables [63]. Depending on the characteristics of the datasets and the results of prior work [28], we determine the datasets that require deseasonalisation (Section 4.1). If required, deseasonalisation is performed by applying the Seasonal and Trend Decomposition using Loess (STL) method [68], and the deseasonalised series are used to train the RNN model. For the datasets that do not require deseasonalisation, the preprocessed series are directly used to train the RNN model, along with the seasonal regressors. In particular, for multiseasonal datasets, we use Fourier terms [69], a set of sine and cosine terms, together with the preprocessed series when training the RNN model following the procedure suggested by Bandara et al. [63]. The `stl` and `fourier` functions in the `forecast` package [70] are used, respectively, for deseasonalisation and Fourier terms calculation. Finally, the moving window scheme is applied to the preprocessed series to create the input and output window pairs required to train the NN models [28]. On the other hand, when implementing the PR models, only mean normalisation is used to preprocess the time series. This is because, unlike in NNs, PR models do not have activation functions, and therewith do not show

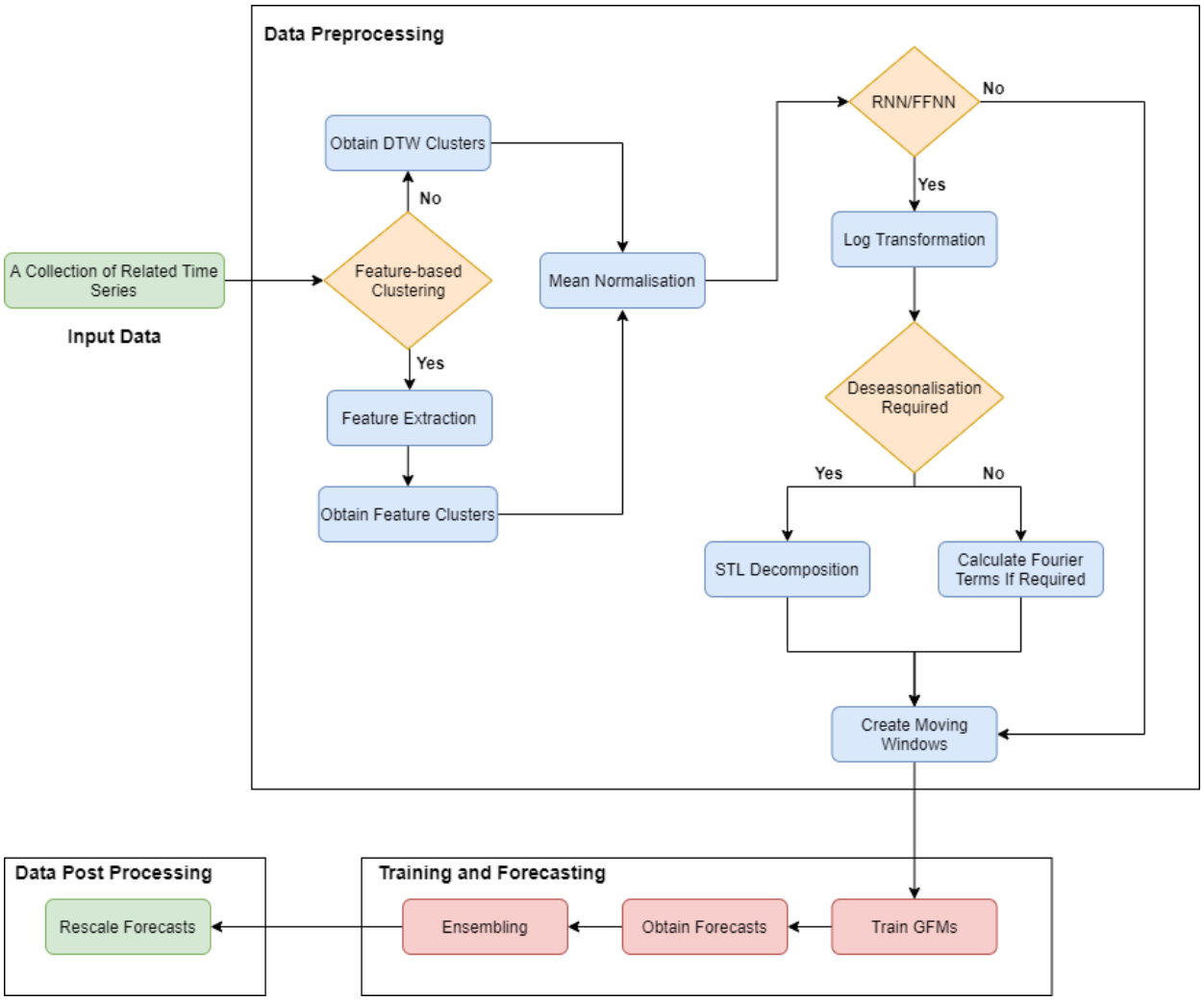

Figure 2: Overall workflow of the proposed ensembled GFMs. The data are first preprocessed based on the GFM type. The preprocessed data are then used to train multiple GFMs whereas their forecasts are ensembled accordingly and rescaled to obtain the final forecasts.

saturation effects in the way NNs do [28]. Once the forecasts are obtained from the GFMs, to bring them to the original scale, the corresponding preprocessing techniques are reversely applied. Figure 2 shows the overall workflow of the proposed ensembled GFMs.

3.3. Ensemble Models to Localise Global Forecasting Models

In the following, we describe the different types of GFM-based ensemble models used in our experiments.

3.3.1. Ensemble of Specialists/Experts

The ensemble of specialists was proposed by Smyl [10], and used by the same author in ES-RNN, the winning solution of the M4 forecasting competition [2]. The ES-RNN implementation uses RNN with LSTM cells as the primary specialist/expert prediction unit.

Algorithm 1 Ensemble of Specialists Algorithm

Parameters
training_set - Contains the training parts of each series of the dataset
validation_set - Contains the training and validation parts of each series of the dataset
test_set - Contains all full-length time series
validation_results - Contains actual data corresponding with the validation parts of each series
num_of_specialists - Number of GFMs to be used in the algorithm
N - Number of top GFMs whose forecasts are combined to obtain the final forecasts

procedure ENSEMBLE_OF_SPECIALISTS($training\_set$, $validation\_set$, $test\_set$, $validation\_results$, $num\_of\_specialists$, N)
    $gfms \leftarrow []$ **Stores GFM specialists**
    $train\_series \leftarrow []$ **Stores training series corresponding with each GFM**
    $val\_predictions \leftarrow []$ **Stores validation forecasts given by GFMs for a particular iteration**
    $test\_predictions \leftarrow []$ **Stores GFM forecasts corresponding with testing period**
    $final\_predictions \leftarrow []$ **Stores final output forecasts**
    $val\_errors \leftarrow []$ **Stores validation errors given by each GFM in a particular iteration**
    $prev\_errors \leftarrow []$ **Stores validation errors given by GFMs in the previous iteration**
    $val\_error\_growing \leftarrow FALSE$ **Whether the validation set error is growing or not**
    $iterations \leftarrow 1$
    for i in 1 to $num\_of\_specialists$ **do**
        $gfms[i] \leftarrow init\_gfm\_model()$
        $train\_series[i] \leftarrow random(training\_set, 0.5)$ **Randomly allocate training series for GFMs**
    end for
    while $val\_error\_growing == FALSE$ **do**
        if $iterations > 1$ **then**
            Re-assign series to their corresponding top performing GFMs
            $train\_series \leftarrow reassign\_series(val\_errors, N)$
        end if
        for i in 1 to $num\_of\_specialists$ **do**
            $gfm[i] \leftarrow train\_gfm(gfm[i], train\_series[i])$ **Train GFMs**
            $val\_predictions[i] \leftarrow test\_gfm(gfm[i], validation\_set)$ **Get validation forecasts**
            Calculate validation errors
            $val\_errors[i] \leftarrow calc\_errors(val\_predictions[i], validation\_results)$
        end for
        if $iterations > 1$ **then**
            Check whether the average validation error is growing or not
            $val\_error\_growing \leftarrow check\_error\_growing(val\_errors, prev\_errors)$
        end if
        $prev\_errors \leftarrow val\_errors$
        $iterations \leftarrow iterations + 1$
    end while

Get forecasts corresponding with the testing period from trained GFMs
for i in 1 to *specialists* **do**
$test\_predictions[i] \leftarrow test\_gfm(gfm[i], test\_set)$
end for
The final forecasts of each series are obtained by averaging the forecasts provided by its top performing GFMs
for s in 1 to len($test\_set$) **do**
$final\_predictions[s] \leftarrow combine\_best\_predictions(test\_predictions, N, s)$
end for
return $final\_predictions$
end procedure

Algorithm 1 shows the training and testing phases of the ensemble of specialists method. In the training phase, the algorithm learns a group of GFMs simultaneously to obtain the best GFMs to model each time series. In the first iteration, 50% of the time series are randomly allocated for each GFM to train. Then, each GFM is trained using its assigned series and the forecasts are obtained from the trained GFMs for a validation dataset. The GFMs are then ranked for each time series based on the validation errors and the time series are re-allocated for their corresponding top N (in our case $N = 2$) GFMs. The series allocation procedure is repeated for each GFM until the average validation error starts to grow. In the testing phase, for each test series, the average of the forecasts corresponding to the top N GFMs forms the final forecast. We tune the number of specialists (K) as a hyperparameter using a validation dataset (refer to Section 4.3).

3.3.2. Clustering-based Ensemble Models

In this approach, we use a combination of feature and distance-based clustering techniques to build ensemble models. Table 1 summarises the clustering techniques used in our experiments. Here, the Category column indicates the approach to which the clustering algorithm is applied, and the Package and Function columns provide a reference to the software implementation used in our experiments.

Clustering Technique	**Category**	**Package**	**Function**
K-means	feature-based	stats [71]	kmeans
K-means++	feature-based	LICORS [72]	kmeanspp
K-medoids with DTW Distance	distance-based	dtwclust [73]	tsclust

Table 1: Overview of Clustering Techniques

For the feature-based clustering approach, we use K-means [21] and K-means++ [22] as the primary clustering algorithms. Here, the K-means++ algorithm is used to address the weak centroid initialisation problem present in the traditional K-means algorithm. We extract a set of self-explainable features [74] from each time series and cluster them to identify similar groups of time series. This approach is similar to the methodology suggested by

Bandara et al. [12]. The extracted features include mean, variance, first order of autocorrelation, trend, linearity, curvature, entropy, lumpiness, spikiness, level shift, variance change, flat spots, crossing points, maximum Kullback-Leibler (KL) shift and time KL shift. We use the `tsfeatures` function from the `tsfeatures` package [75] to extract these time series features. In this way, we represent each time series as a vector of features. We then apply the target clustering algorithms to determine the optimal feature grouping, which represents the subgroups of time series that are used to train the GFMs. There have been other feature extraction procedures proposed in the literature, such as catch22 [76] or tsfresh [77], and their use would most likely result in different clustering outcomes. However, `tsfeatures` is a popular and well-established package in the forecasting space, and has been used in a similar setting by Bandara et al. [12], which is why we focus on this package.

To implement the distance-based clustering approach, we use the K-medoids clustering [23] algorithm, using DTW as the primary distance measure. The K-medoids algorithm handles noise and outliers better compared with K-means. As distance-based clustering algorithms work directly on the raw observations of the time series, a prior preprocessing phase is not necessary.

Based on the above clustering approaches, we introduce the following model variants to our framework:

GFM.Cluster.Number We apply clustering algorithms multiple times by varying the number of clusters. Multiple GFMs are then trained per each identified cluster, for multiple iterations, generating multiple forecasts per each series. These forecasts are then averaged to obtain the final forecasts.

GFM.Cluster.Seed We use the elbow method [52] to determine the optimal number of clusters, and then apply the K-means or K-means++ algorithms multiple times with the optimal number of clusters by changing the cluster seed. Finally, multiple GFMs are trained per each cluster, for multiple iterations and the forecasts are ensembled following the same procedure used in GFM.Cluster.Number.

To determine the optimal number of clusters, the elbow method runs a clustering algorithm multiple times by gradually increasing the number of clusters. It calculates the sum of squared errors in clustering for each considered number of clusters and plots the errors against their respective number of clusters. At the beginning, the error is expected to quickly decrease as the number of clusters increases. At some point, the error reduction rate slows down leading to a situation of diminishing returns. This point can be determined as the point of maximum curvature (forming an elbow) in the plot and the number of clusters corresponding with that point is considered as the optimal number of clusters to be used with the considered clustering algorithm. We visualise the results of the elbow method with our experimental datasets using the R package `factoextra` [78].

Figure 3 illustrates the GFM.Cluster.Number variant used in our experiments, and Algorithms 2 and 3 show the training and testing phases of the GFM.Cluster.Number and

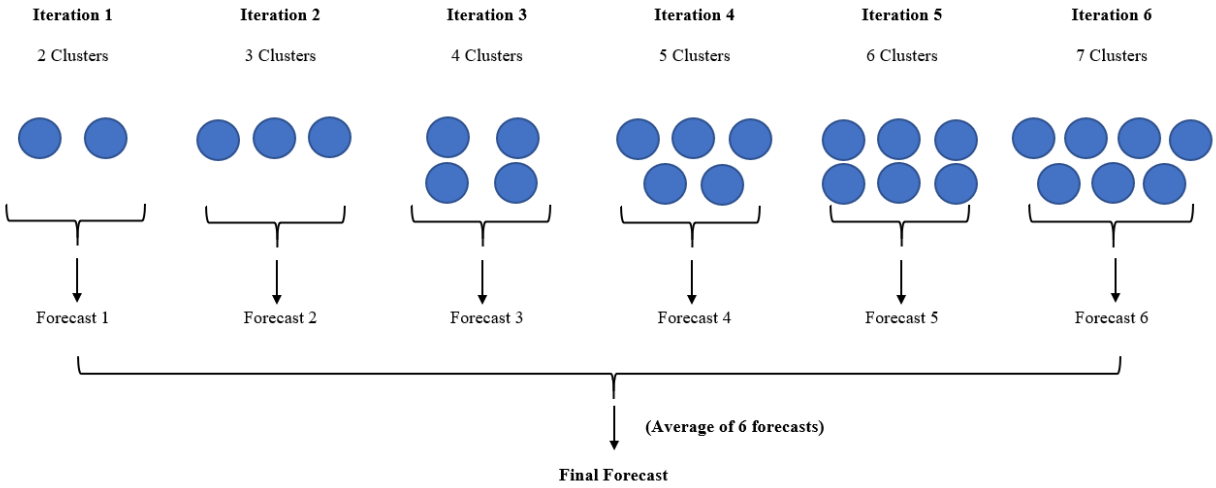

Figure 3: In the GFM.Cluster.Number variant, we consider the cluster range from two to seven. Six forecasts are generated for each time series by varying the cluster number. Finally, these forecasts are averaged to obtain the final forecasts. On the other hand, the GFM.Cluster.Seed variant uses the same number of clusters, the optimal cluster number, varying the clusters from iteration to iteration by varying the cluster seeds.

GFM.Cluster.Seed models, respectively. We note that the range of cluster numbers for GFM.Cluster.Number is defined in a way that the optimal number of clusters found by the elbow method is placed within that range.

The feature clustering technique is applied for both the GFM.Cluster.Number and GFM.Cluster.Seed variants, whereas the distance-based clustering technique is only applied for the GFM.Cluster.Number variant. This is because, the GFM.Cluster.Seed algorithm requires an optimal number of clusters to operate. According to the best of our knowledge, there is no straightforward method to identify the optimal number of clusters for K-medoids clustering with DTW distances. So, altogether there are five clustered ensemble models that we use in the experiments: K-means, K-means++ and K-medoids (DTW) with the GFM.Cluster.Number variant, and K-means and K-means++ with the GFM.Cluster.Seed variant.

3.3.3. Forecast Combinations of Global and Local Forecasting Models

We combine the predictions of six RNN-based GFMs with the predictions of two traditional univariate forecasting models. The RNN-based models are a single RNN and five clustered ensemble models based on RNNs, namely: RNN.Cluster.Number with K-means, K-means++ and K-medoids, and RNN.Cluster.Seed with K-means and K-means++ methods. The traditional techniques are ETS [4] and ARIMA [5]. The RNN-based forecasts are averaged with ETS forecasts as well as with both ETS and ARIMA forecasts, making 12 forecast combination models.

When applying forecast combinations to multi-seasonal benchmark datasets, we replace the ETS and ARIMA methods with the Trigonometric, Box-Cox, ARMA, Trend, Seasonal [TBATS, 79] and the Dynamic Harmonic Regression ARIMA [DHR-ARIMA, 80] methods,

Algorithm 2 GFM.Cluster.Number Algorithm

Parameters
training_set - Contains the training parts of each series of the dataset
test_set - Contains all full-length time series
cluster_range - A vector of clusters numbers
clustering_method - The clustering technique e.g. K-means

procedure ENSEMBLE_OF_CLUSTERING_GFM.CLUSTER.NUMBER(*training_set*, *test_set*, *cluster_range*, *clustering_method*)
    predictions ← [][] **Stores forecasts of all iterations**
    final_predictions ← [] **Stores final average forecasts**
    iterations ← 1
    seed ← *generate_seed*()
    for *cluster_num* in *cluster_range* **do**
        Cluster all series for the numbers specified in cluster_range iteratively
        cluster_ids ← *cluster_series*(*test_set*, *cluster_num*, *clustering_method*, *seed*)
        for *c* in *cluster_num* **do**
            series_ids ← *cluster_ids*[*c*]
            gfm ← *init_gfm_model*()
            gfm ← *train_gfm*(*gfm*, *train_series*[*series_ids*]) **Train GFMs using clustered series**
            Obtain forecasts for the test series in clusters using the trained GFMs
            predictions[*series_ids*][*iterations*] ← *test_gfm*(*gfm*, *test_series*[*series_ids*])
        end for
        iterations ← *iterations* + 1
    end for
    Calculate the average of forecasts produced in each iteration
    final_predictions ← *row_wise_avg*(*predictions*)
     return *final_predictions*
end procedure

Algorithm 3 GFM.Cluster.Seed Algorithm

```
Parameters
training_set - Contains the training parts of each series of the dataset
test_set - Contains all full-length time series
optimal_num_of_clusters - The optimal number of clusters to be used given by the elbow method
num_of_iterations - Number of cluster generating iterations
clustering_method - The clustering technique e.g. K-means

procedure ENSEMBLE_OF_CLUSTERING_GFM.CLUSTER.SEED(training_set, test_set, optimal_num_of_clusters, num_of_iterations, clustering_method)
    predictions ← [][] Stores forecasts of all iterations
    final_predictions ← [] Stores final average forecasts
    for iter in num_of_iteration do
        seed ← generate_seed() Generate a random seed in each iteration
        Cluster all series into the optimal number of clusters with the generated seed
        cluster_ids ← cluster_series(test_set, optimal_num_of_clusters, clustering_method, seed)
        for c in optimal_num_of_clusters do
            series_ids ← cluster_ids[c]
            gfm ← init_gfm_model()
            gfm ← train_gfm(gfm, train_series[series_ids]) Train GFMs using clustered series
            Obtain forecasts for the test series in clusters using the trained GFMs
            predictions[series_ids][iter] ← test_gfm(gfm, test_series[series_ids])
        end for
    end for
    Calculate the average of forecasts produced in each iteration
    final_predictions ← row_wise_avg(predictions)
     return final_predictions
end procedure
```

which are capable of forecasting time series with multiple seasonal patterns.

3.3.4. Ensemble Baseline

Ensembling can be used in many different ways with different objectives. E.g., boosting lowers bias, and bagging lowers variance, by addressing model, parameter, and/or data uncertainty. Therefore, we employ an ensembling baseline to be able to quantify how much accuracy is gained from localisation versus from other effects. In particular, we implement a straightforward approach to address the parameter uncertainty of GFMs by training GFMs multiple times using different seeds. We train global RNN/FFNN models using all the available time series with five different seeds. The forecasts generated by these five seeds are then averaged to obtain the final forecasts.

4. Experimental Framework

In this section, we discuss the benchmark datasets, hyper-parameter optimisation approaches, error metrics, and benchmarks used in our experiments.

4.1. Datasets

We use eight publicly available datasets[1] to evaluate the performance of our proposed variants. Table 2 provides summary statistics of these benchmark datasets, and a brief overview is as follows:

- M3 Dataset [81]: Monthly dataset from the M3 forecasting competition. This dataset contains six subcategories of time series, namely micro, macro, industry, demo, finance, and other. When executing our proposed variants, we use this information as the domain knowledge and a separate global model is trained for each subcategory.
- M4 Monthly Dataset [2]: Monthly dataset from the M4 forecasting competition. This dataset contains the same six subcategories as the M3 dataset, and the GFMs are built per subcategory accordingly.
- CIF 2016 Dataset [82]: Monthly dataset from the CIF 2016 forecasting competition, containing 24 time series from the banking domain and 48 simulated series. We use all simulated and real-world series to train models and to evaluate their performance with this dataset. Hence, the simulated series also contribute to the final forecasting accuracy of the dataset.
- Kaggle Wikipedia Web Traffic Dataset [83]: A subset of time series from the Kaggle Wikipedia Web Traffic forecasting competition that contains numbers of daily hits/traffic for a given set of Wikipedia web pages. The missing observations of this dataset are treated as zeros, following the procedure suggested by Hewamalage et al. [28].

[1]The experimental datasets are available at https://drive.google.com/drive/folders/16xqLEFyLn_gJcXrIp1LWyAD_KvJjB5Hn?usp=sharing.

Dataset Name	No. of Time Series	Forecast Horizon	Frequency	Minimum Length	Maximum Length
M3	1428	18	Monthly	48	126
M4	48000	18	Monthly	42	2794
CIF 2016	72	6, 12	Monthly	22	108
Wikipedia Web Traffic	997	59	Daily	550	550
Ausgrid Half-Hourly	300	96	Half-Hourly	1488	1488
Ausgrid Monthly	1405	12	Monthly	38	88
Tourism	427	8	Quarterly	22	122
Hospital	767	12	Monthly	72	72

Table 2: Datasets Information

- Ausgrid Half-Hourly Dataset [84]: A half-hourly time series dataset, representing the energy consumption of Australian households. In this dataset, time series may exhibit multiple seasonal patterns, such as hourly, daily, and weekly seasonality.
- Ausgrid Monthly Dataset [84]: A collection of monthly time series, representing the energy consumption of Australian households. In our experiments, we only include time series with more than 49 observations, to avoid possible constraints in our hyper-parameter tuning process.
- Tourism Quarterly Dataset [85]: Quarterly dataset from a tourism forecasting competition.
- Hospital Dataset: A collection of monthly time series, representing patient counts related to medical products. This dataset was extracted from the R package `expsmooth` [86].

4.2. Dataset Specific Preprocessing

The input window size of NNs or the number of lagged values used in the PR models are determined using the heuristic suggested by Hewamalage et al. [28]. Furthermore, we deseasonalise the M3, M4, CIF 2016 and Kaggle web traffic time series datasets to train RNN models, following the results of Hewamalage et al. [28]. For the Ausgrid monthly, Ausgrid half-hourly dataset, Tourism and Hospital datasets, we run preliminary experiments with different methods to address the seasonality, and as a result deseasonalise the Tourism and Hospital series, do not deseasonalise the Ausgrid monthly series, and use Fourier terms to handle multiple seasonalities in the Ausgrid half-hourly dataset.

4.3. Hyperparameter Tuning

The base learners RNN and FFNN have various hyper-parameters. The RNN models include number of epochs, epoch size, mini-batch size, cell dimension, number of hidden layers, L2 regularisation weights, standard deviation of random normal initialiser and standard deviation of Gaussian noise [28]. The COntinuous COin Betting [COCOB, 87] algorithm

Dataset	RNN								FFNN		Common
	Batch Size	Epochs	Epoch Size	Std. Noise	L2 Reg.	Cell Dim.	Layers	Std. Initializer	Nodes	Decay	Specialists
M3(Mic)	40 - 80	2 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M3(Mac)	30 - 50	2 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M3(Ind)	30 - 50	2 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M3(Dem)	10 - 20	2 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M3(Fin)	10 - 25	2 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M3(Oth)	5 - 10	2 - 30	5 - 20	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M4(Mic)	1000 - 1500	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M4(Mac)	1000 - 1500	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M4(Ind)	1000 - 1500	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M4(Dem)	850 - 950	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M4(Fin)	1000 - 1500	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
M4(Oth)	30 - 40	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 25	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
CIF (12)	5 - 10	2 - 25	5 - 20	0.01 - 0.08	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
CIF (6)	2 - 5	2 - 30	5 - 15	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 5	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
Wikipedia	100 - 150	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 25	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
Ausgrid (HH)	20 - 50	10 - 40	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
Ausgrid (M)	20 - 100	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
Tourism	20 - 100	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7
Hospital	20 - 100	2 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	1 - 12	0 - 0.1	2 - 7

Table 3: Initial Hyperparameter Ranges

is used as the learning algorithm to train the RNN-based variants, which does not require tuning of the learning rate. The FFNN models have number of nodes in the hidden layers and learning rate decay as model hyper-parameters. Furthermore, the number of specialists required by the ensemble of specialists model is also tuned as a separate hyper-parameter for each base learner. Meanwhile, the PR-based model variants do not have hyper-parameters that require tuning. We autonomously determine the optimal values of the hyper-parameters of our base models using the sequential model-based algorithm configuration (SMAC), a variant of Bayesian Optimisation proposed by Hutter et al. [88]. In our experiments, the Python implementation of SMAC [89] is used. The initial hyperparameter ranges used in the SMAC algorithm for each dataset are shown in Table 3. The SMAC algorithm then finds the optimal values for each hyperparameter within the defined range using a predefined number of iterations or a time limit. We use a sequence of observations from the end of each series of length equivalent to the intended forecast horizon as the validation set for hyperparameter tuning. The GFMs are trained using the remaining observations of each series and the forecasts corresponding to the validation set are used to optimise the hyperparameters.

In general, the hyperparameter optimisation process is conducted using all the time series in a dataset. However, our ensemble models use different groups of time series to train multiple submodels and hence, performing separate hyperparameter tuning for each submodel is not efficient. To overcome this issue, a set of time series is randomly chosen from a given dataset with a size of the number of series divided by the maximum number of submodels considered in one iteration of an ensemble model, e.g., the maximum number of submodels used in one iteration of the ensemble of specialists or clustered ensemble models. The hyperparameters are tuned based on this chosen set of time series. The same set of hyperparameters are then used to train each submodel in the considered ensemble models. We also train non-ensemble models using the same set of hyperparameters, for a consistent comparison against ensemble models.

4.4. Error Metrics

We measure the performance of our models using the symmetric Mean Absolute Percentage Error (sMAPE), Mean Absolute Scaled Error [MASE, 90], Mean Absolute Error [MAE, 23] and Root Mean Squared Error (RMSE) which are commonly used error metrics in the field of time series forecasting. Equations 1, 2, 3 and 4 define the sMAPE, MASE, MAE and RMSE error metrics, respectively. Here, M is the number of instances in the training set, N is the number of data points in the test set, F_k are the forecasts, Y_k are the actual values for the required forecast horizon and S is the length of the seasonal cycle.

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

$$MASE = \frac{\sum_{k=1}^{N} |F_k - Y_k|}{\frac{N}{M-S} \sum_{k=S+1}^{M} |Y_k - Y_{k-S}|} \tag{2}$$

$$MAE = \frac{\sum_{k=1}^{N} |F_k - Y_k|}{N} \tag{3}$$

$$RMSE = \sqrt{\frac{\sum_{k=1}^{N} |F_k - Y_k|^2}{N}} \tag{4}$$

For datasets containing zeros, namely the Ausgrid datasets and the Kaggle web traffic dataset, we use a variant of the sMAPE error measure proposed by Suilin [91], to avoid possible division by zero problems. To achieve this, the denominator of Equation 1 is changed to $max(|Y_k| + |F_k| + \epsilon, 0.5 + \epsilon)$, where we set ϵ to its proposed default value of 0.1.

Since all these error measures are defined for each time series, we calculate the mean and median values of them across a dataset to measure the model performance. Therefore, each model is evaluated using eight error metrics: mean sMAPE, median sMAPE, mean MASE, median MASE, mean MAE, median MAE, mean RMSE and median RMSE.

4.5. Benchmarks and Variants

To compare against our proposed ensemble variants, we use two univariate forecasting models, ETS [4] and ARIMA [5], which are competitive and commonly used benchmarks in time series forecasting research. Furthermore, we use TBATS [79] and DHR-ARIMA [80] as the univariate benchmarks for the Ausgrid half-hourly dataset, as ETS and ARIMA are not well-suited to handle the multiple seasonal patterns present in half-hourly data. We use the default implementations available from the `forecast` package of these methods. Furthermore, we use a grid-search approach to determine the optimal number of Fourier terms for DHR-ARIMA, which controls the smoothness of the seasonal pattern to be modelled.

We also use 2 deep learning benchmarks, DeepAR [26] and N-BEATS [27]. DeepAR is a global auto-regressive RNN used with probabilistic forecasting. N-BEATS is an interpretable deep neural network based forecasting framework which uses backward and forward residual links with a deep stack of fully-connected layers. In our work, we use the implementations of

DeepAR and N-BEATS available from the Python package `GluonTS` [92] with their default hyperparameters.

As a benchmark based on gradient boosted trees, we use CatBoost [93]. In contrast to other popular implementations of gradient boosted trees, it considers the order of the data points during model training, thus making it more appropriate for time series forecasting. In our work, we use the implementation of CatBoost available from the R package `catboost` [93] with its default hyperparameters that are expected to provide good results.

As an ensemble benchmark, we use FFORMA, which combines the forecasts from a set of univariate models based on the weights obtained by training a gradient boosted tree model using 42 features, which are calculated from the original time series. This method achieved the overall second place in the M4 forecasting competition [49]. In our work, we use the implementation of FFORMA available from the `M4metalearning` package [94]. Apart from these, the following benchmarks are used in our experiments.

Baseline The forecasts provided by a single FFNN, RNN, or PR model trained across a full dataset.

Kmeans.OC Use K-means clustering with optimal number of clusters to cluster the features, then train separate GFMs per cluster, so that forecasts for a series are obtained by one particular GFM.

KmeansPlus.OC The same as Kmeans.OC, but using the K-means++ clustering algorithm.

Xmeans The same as the last two methods, but with Xmeans clustering [95], which is able to automatically identify an optimal number of clusters. We use the `XMeans` function in the `RWeka` package [96] to implement Xmeans clustering.

Furthermore, we benchmark the proposed methods against random clustering methods, as suggestsed in Montero-Manso and Hyndman [6]. Random clustering methods are proposed as a way to increase model complexity. In particular, we use three models based on random clustering as benchmarks, as follows:

Random.OC Cluster series randomly, train separate GFMs with each series cluster and obtain forecasts for each series once with its corresponding GFM.

Random.Number Cluster series randomly and build the GFM.Cluster.Number model.

Random.Seed Cluster series randomly and build the GFM.Cluster.Seed model.

The five variants of the clustered ensemble models (Section 3.3.2), ensemble of specialists and ensemble of NNs are used as follows:

DTW.Number Apply K-medoids clustering with DTW distances for series and build GFM.Cluster.Number model.

Kmeans.Number Apply K-means clustering for features and build the GFM.Cluster.Number model.

KmeansPlus.Number Apply K-means++ clustering for features and build the GFM.Cluster.Number model.

Kmeans.Seed Apply K-means clustering for features and build the GFM.Cluster.Seed model.

KmeansPlus.Seed Apply K-means++ clustering for features and build the GFM.Cluster.Seed model.

Ensemble.Specialists Ensemble of Specialists using RNNs, FFNNs and PR as the base models.

Ensemble.Seed Train NNs with different seeds in multiple iterations and average the results.

4.6. Statistical Testing of the Results

We consider the recommendations given by Demsar [97] to analyse the statistical significance of results provided by different forecasting methods across the experimental datasets. The null hypothesis of no difference between the methods is tested by applying a Friedman test [98], and if the null hypothesis is rejected, a pairwise statistical comparison: the Wilcoxon signed-rank test with Holm's alpha correction ($\alpha = 0.05$) is used following the recommendations given by Benavoli et al. [99]. The results of statistical tests are visualised using a critical difference diagram [97]. We consider ETS and TBATS, and ARIMA and DHR-ARIMA as pairs together for statistical testing as we do not evaluate these models with each dataset. All other 45 models including FFORMA, DeepAR, N-BEATS, CatBoost and GFMs based on RNNs, FFNNs, and PR are individually considered for statistical testing. We do not consider the M4 monthly dataset for statistical testing, as due to its much large size compared with the other datasets it would dominate the results. Finally, the mean sMAPE results of 47 models across the other 7 experimental datasets are used for statistical testing. We use the Python implementation `cd-diagram` available on Github to plot the critical difference diagram [100].

5. Analysis of Results

This section provides a comprehensive analysis of the results of all considered models across the experimental datasets. Table 4 summarises the overall performance of the proposed variants and the benchmarks, in terms of mean sMAPE across all experimental datasets. The conclusions we draw by analysing the mean sMAPE values are in agreement with the other seven error metrics that are available in the Online Appendix[2].

[2]The online appendix is available at https://drive.google.com/file/d/1lJuGNvGA8O-UOOSh8KHUEPeNvhESv75W/view?usp=sharing.

Model	M3	M4	CIF	Ausgrid HH	Ausgrid M	Kaggle	Tourism	Hospital
ETS	**14.14**	13.53	12.18	-	26.49	52.54	15.07	17.46
ARIMA	14.25	13.08	**11.70**	-	26.69	**47.96**	16.58	17.79
TBATS	-	-	-	**36.20**	-	-	-	-
DHR-ARIMA	-	-	-	39.31	-	-	-	-
FFORMA	14.51	**12.64**	13.23	46.40	26.30	48.65	14.88	**17.13**
CatBoost	16.41	14.05	14.87	44.07	24.92	51.50	16.53	18.09
DeepAR	15.74	14.29	13.58	68.47	**24.75**	57.79	15.29	17.45
N-BEATS	14.76	13.40	11.72	44.89	45.08	51.82	**14.45**	17.77
RNN Baseline	14.82	14.21	11.47	34.96	26.76	46.52	**15.99**	19.80
RNN Kmeans.OC	14.49	13.90	11.14	35.27	27.37	47.49	16.32	19.45
RNN KmeansPlus.OC	14.56	14.18	11.17	35.54	25.64	47.39	16.28	19.48
RNN Xmeans	14.27	13.65	10.69	35.27	25.69	47.69	16.44	19.50
RNN DTW.Number	**14.14**	**13.48**	11.10	**34.79**	25.64	46.99	16.46	19.40
RNN Kmeans.Number	14.33	13.65	10.77	34.83	25.23	46.88	16.39	**19.30**
RNN KmeansPlus.Number	14.32	13.64	10.92	34.88	**25.04**	47.35	16.37	19.35
RNN Kmeans.Seed	14.40	14.05	11.14	34.82	26.92	47.05	16.31	19.39
RNN KmeansPlus.Seed	14.48	13.99	11.17	34.91	25.78	46.88	16.26	19.44
RNN Ensemble.Specialists	14.55	13.96	**10.55**	34.92	25.56	46.63	16.25	19.50
RNN Ensemble.Seed	14.74	14.39	10.98	34.94	26.72	**46.42**	16.21	19.76
RNN Random.OC	14.78	14.79	10.91	36.91	25.58	46.91	16.50	19.34
RNN Random.Number	14.64	14.04	10.81	35.28	25.69	47.03	16.49	19.41
RNN Random.Seed	14.59	14.82	10.84	35.49	25.25	47.14	16.24	19.34
FFNN Baseline	15.60	15.75	16.30	38.72	25.03	47.69	16.49	18.76
FFNN Kmeans.OC	15.71	14.96	13.54	45.04	25.79	48.00	18.30	18.10
FFNN KmeansPlus.OC	15.19	14.49	13.29	41.80	25.32	47.76	17.86	18.48
FFNN Xmeans	15.26	14.28	14.29	40.03	25.00	47.55	18.57	18.15
FFNN DTW.Number	**14.89**	**13.75**	13.44	39.79	**24.97**	47.75	16.79	**17.61**
FFNN Kmeans.Number	15.18	14.15	14.06	37.91	25.38	47.57	17.36	17.92
FFNN KmeansPlus.Number	15.02	14.15	13.80	37.87	25.38	**47.52**	17.64	17.98
FFNN Kmeans.Seed	15.57	14.50	**13.11**	38.19	25.39	47.62	18.06	17.90
FFNN KmeansPlus.Seed	15.37	14.50	13.52	38.21	25.37	47.56	17.63	18.07
FFNN Ensemble.Specialists	15.61	14.46	15.03	39.34	24.99	47.80	**15.70**	18.12
FFNN Ensemble.Seed	15.36	14.88	15.06	**37.36**	25.00	47.81	16.49	18.36
FFNN Random.OC	15.79	14.95	15.69	38.51	25.24	47.72	16.77	18.63
FFNN Random.Number	15.42	14.55	14.82	39.14	25.15	47.82	16.51	18.38
FFNN Random.Seed	15.39	14.62	14.40	37.62	25.21	47.88	16.42	18.27
PR Baseline	14.36	13.77	12.86	39.81	24.08	131.90	15.86	17.56
PR Kmeans.OC	14.29	13.30	12.24	39.74	24.24	90.00	15.34	17.82
PR KmeansPlus.OC	14.22	23.12	**12.19**	39.82	24.27	90.00	15.43	17.90
PR Xmeans	14.31	13.04	12.79	39.72	**24.04**	90.01	15.45	17.89
PR DTW.Number	14.04	**12.87**	12.49	39.56	24.20	103.37	15.59	17.87
PR Kmeans.Number	**13.91**	13.00	12.33	39.71	24.06	91.48	15.16	**17.43**
PR KmeansPlus.Number	14.03	13.00	12.60	39.96	24.09	**88.73**	15.17	17.56
PR Kmeans.Seed	14.09	13.29	12.21	39.75	24.24	90.00	15.34	17.87
PR KmeansPlus.Seed	14.22	13.30	12.24	39.83	24.27	90.00	15.34	17.90
PR Ensemble.Specialists	14.09	13.63	12.47	**39.47**	24.28	116.09	**15.06**	17.54
PR Random.OC	14.41	13.77	12.86	39.86	24.06	131.38	15.85	17.62
PR Random.Number	14.27	13.73	12.64	39.80	24.07	128.56	15.79	17.59
PR Random.Seed	14.24	13.76	12.67	39.81	24.09	127.96	15.84	17.58

Table 4: Mean sMAPE results. The benchmark models, the RNN-based models, the FFNN-based models, and the PR-based models are separated with double lines. The corresponding best-performing model for each section is highlighted in boldface.

5.1. Relative Performance of Clustered Ensemble Models

We compare the relative performance of seven clustered ensemble models: Kmeans.Number, KmeansPlus.Number, Random.Number, DTW.Number, Kmeans.Seed, KmeansPlus.Seed and Random.Seed across all datasets. Figure 4 shows the mean sMAPE ranks of these clustered ensemble models implemented using the 3 base GFMs: RNNs, FFNNs and PR models using violin plots enabling to compare their performance in terms of the inter-quantile ranges and density distributions of error ranking.

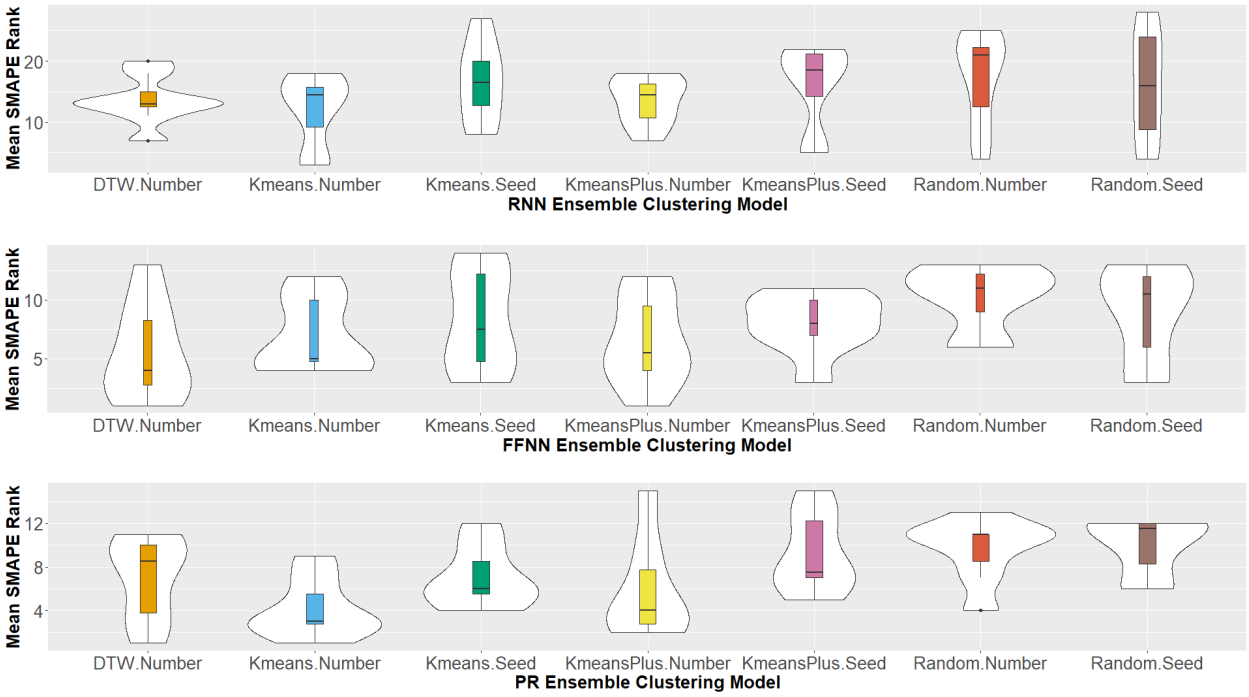

Figure 4: Relative Performance of Clustered Ensemble Models

The clustered ensemble models that correspond to different GFMs show varied performances. According to Figure 4, we see that for the RNN and FFNN models, the DTW.Number cluster variant obtains the best mean sMAPE rank whereas the Kmeans.Number cluster variant obtains the best mean sMAPE rank for the PR models. The inter-quantile ranges of DTW.Number and Kmeans.Number cluster variants are respectively lower across NN and PR models. Furthermore, higher density areas of the DTW.Number and Kmeans.Number cluster variants are respectively located much lower along the y axis across NN and PR models. The results also indicate that overall, the GFM.Cluster.Number based cluster variants outperform the GFM.Cluster.Seed based cluster variants.

With respect to the random clustering models, overall, Random.Number and Random.Seed perform worst since their inter-quantile ranges and density distributions are located much higher along the y-axis compared to other variants. Hence, our experiments

clearly show that increasing the model complexity by using sophisticated clustering techniques is able to improve the forecasting accuracy.

5.2. Relative Performance of Clustered Non-Ensemble Models

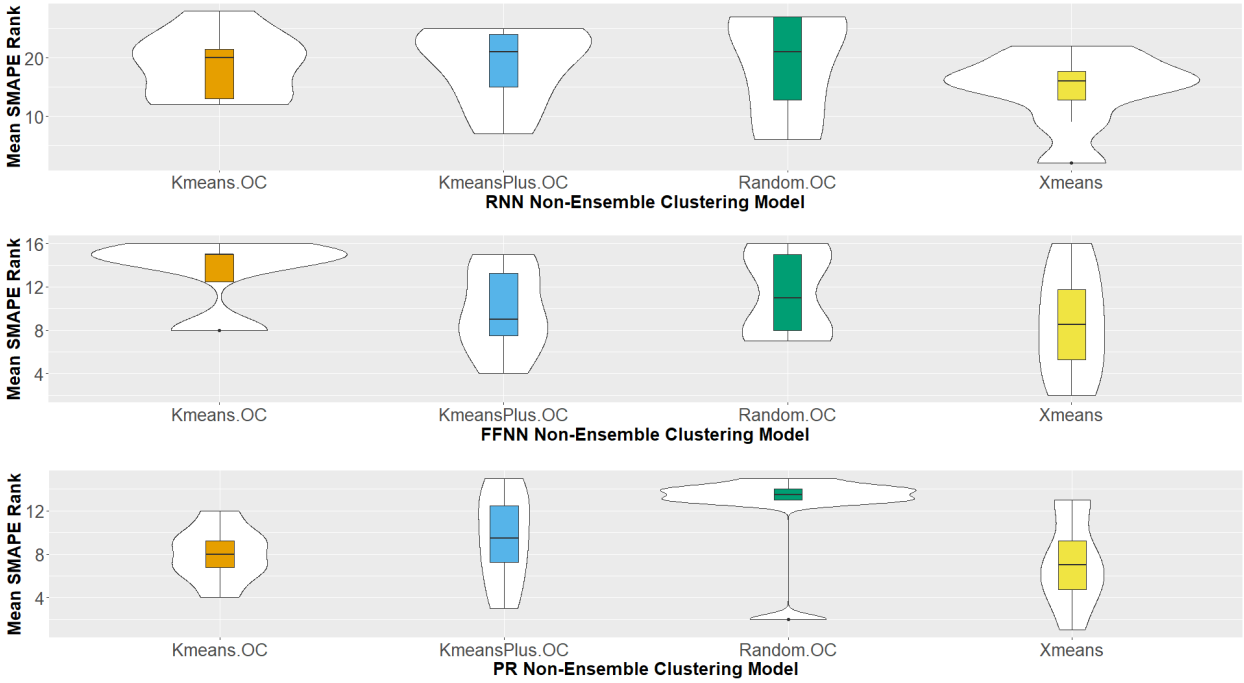

Figure 5: Relative Performance of Clustered Non-Ensemble Models

Figure 5 shows the relative performance of clustered non-ensemble models: Kmeans.OC, KmeansPlus.OC, Random.OC and Xmeans based on mean sMAPE ranks for the 3 base GFMs. Xmeans shows the best performance over all GFMs. Random.OC shows the worst performance with RNNs and PR models, whereas Kmeans.OC shows the worst performance with FFNNs since the higher density areas of those variants are located much higher across the y axis.

Xmeans identifies the optimal number of clusters to be used by itself and therefore, it is capable of reducing the prediction error compared to Kmeans.OC and KmeansPlus.OC where the optimal number of clusters is determined using the elbow method [52] with a graphical approach.

5.3. Performance of Clustered Ensemble Models Vs. Clustered Non-Ensemble Models

In Table 4 we see that our proposed five clustered ensemble variants: DTW.Number, Kmeans.Number, KmeansPlus.Number, Kmeans.Seed, and KmeansPlus.Seed show a better performance compared to clustered non-ensemble models across all the datasets with all GFMs except for 3 cases: RNN Xmeans and PR KmeansPlus.OC on the CIF 2016 dataset, PR Xmeans on the Ausgrid monthly dataset.

In non-ensemble clustering, information loss can happen due to training multiple models with different subgroups of datasets. Ensembling of the results of clustering models in multiple iterations helps to mitigate the information loss while properly addressing data heterogeneity during model training. With this, the methods can provide more accurate forecasts than the clustered non-ensemble methods.

5.4. Performance of Non-Clustering based Ensemble Models and Deep Learning Benchmarks

From Table 4, we can furthermore see that the Ensemble.Specialists variant shows an overall better performance compared to the Baseline models across all GFMs. However, our feature and distance-based clustered ensemble models outperform the Ensemble.Specialists for all GFMs across all benchmark datasets, except for 6 cases: CIF 2016, Kaggle web traffic and Tourism on RNNs, Tourism on FFNNs and Ausgrid half-hourly and Tourism on PR. Although the ensemble of specialists considers data heterogeneity during model training, it does not outperform the clustered ensemble techniques in many cases. For our experiments, we use a generic version of Ensemble.Specialists. We also note that the original implementation of the Ensemble.Specialists method uses a different base method than we do. Thus, the Ensemble.Specialists variant may require advanced preprocessing techniques, advanced GFM architectures based on datasets and a set of hyperparameter values chosen based on expert knowledge to provide better results, where in our version, the design choices do not depend on the dataset or expert knowledge.

Overall, the Ensemble.Seed variants show a better performance compared to the Baseline models. The RNN Baseline only outperforms the RNN Ensemble.Seed on the M4 and Tourism datasets and the FFNN Baseline only outperforms the FFNN Ensemble.Seed variant on the Kaggle web traffic dataset. As the Ensemble.Seed models address parameter uncertainty they are able to achieve better performance than the Baseline models. However, all other ensemble variants show an overall better performance compared with the Ensemble.Seed models.

The FFORMA method outperforms our clustered ensemble models on the M4, Tourism and Hospital datasets across all GFMs, and shows better performance compared to FFNN and PR on the M3 dataset and the Kaggle web traffic dataset, respectively. However, overall our clustered ensemble models outperform FFORMA. Moreover, Ensemble.Specialists models show a mixed performance compared to FFORMA across all baseline GFMs. The poor performance of the FFORMA method on the Kaggle web traffic and Ausgrid half-hourly datasets can be attributed to its instability around forecasting time series with consecutive zeros/constant values. Also, the limited amount of time series available for model training may be a reason for the underperformance of FFORMA on the CIF 2016 dataset.

Our clustered ensemble models outperform the considered gradient boosted trees based benchmark, CatBoost, for all GFMs across all datasets, except for 5 cases: Ausgrid monthly and Hospital on RNNs, Ausgrid monthly and Tourism on FFNNs and Kaggle web traffic on PR.

Furthermore, our clustered ensemble models overall outperform DeepAR across all datasets and N-BEATS across 7 datasets: M3, M4, CIF 2016, Kaggle web traffic, Ausgrid half hourly, Ausgrid monthly and Hospital.

5.5. Performance of Global Models and Traditional Univariate Forecasting Models

According to the results of Table 4, overall, RNNs show a better performance than FFNNs across all experimental datasets. Furthermore, RNNs and PR-based GFM variants show a competitive performance. RNNs outperform the PR models for the CIF 2016, Kaggle web traffic, and Ausgrid half-hourly datasets, while for the remaining 5 datasets, PR models demonstrate a better performance than RNNs.

We further compare the performance of the traditional univariate forecasting models and the three GFMs: RNNs, FFNNs, and PR models. As shown in Table 4, even the RNN Baseline outperforms the individual univariate forecasting models on the CIF 2016, Ausgrid half-hourly and Kaggle web traffic datasets. The PR Baseline model outperforms the univariate forecasting models on the Ausgrid monthly dataset. For the Tourism dataset, all GFM baselines and for the Hospital dataset, the PR baseline show a better performance than ARIMA. For the M3 and M4 datasets, the traditional univariate benchmarks show a better performance than the Baseline GFMs. But our clustered ensemble models and ensemble of specialist models based on GFMs, outperform the univariate forecasting benchmarks across all benchmark datasets.

Local Models	RNN Variant	M3	M4	CIF	Ausgrid HH	Ausgrid M	Kaggle	Tourism	Hospital
ETS/TBATS	Baseline	13.86	13.10	11.32	34.30	24.18	48.49	14.39	17.79
	DTW.Number	13.67	12.80	11.18	34.27	23.93	49.13	14.50	17.68
	Kmeans.Number	13.68	12.95	**11.03**	34.22	23.83	49.10	14.55	17.65
	KmeansPlus.Number	13.66	12.95	11.07	34.23	23.83	49.37	14.56	17.67
	Kmeans.Seed	13.71	13.02	11.17	34.18	24.14	49.23	14.54	17.70
	KmeansPlus.Seed	13.73	13.03	11.20	**34.12**	24.10	49.10	14.51	17.71
ETS/TBATS, ARIMA/DHR-ARIMA	Baseline	13.54	12.73	11.25	34.75	23.75	**48.07**	**14.26**	17.36
	DTW.Number	13.45	**12.57**	11.21	34.73	23.70	48.49	14.35	17.32
	Kmeans.Number	13.46	12.67	11.10	34.70	**23.69**	48.49	14.40	**17.30**
	KmeansPlus.Number	**13.44**	12.67	11.14	34.70	23.71	48.66	14.42	17.31
	Kmeans.Seed	13.47	12.69	11.21	34.68	23.72	48.57	14.40	17.33
	KmeansPlus.Seed	13.49	12.70	11.22	34.62	23.85	48.49	14.37	17.34

Table 5: Mean sMAPE Results of Forecast Combinations Across All Experimental Datasets

5.6. Performance of Forecast Combinations with Global and Local Models

We compare the performance of 6 RNN-based models: RNN Baseline and 5 feature and distance-based clustered ensemble models against the forecast combinations that combine the forecasts of RNN-based models and traditional univariate forecasting models. Table 5 shows the performance of the forecast combinations compared with the RNN-based models on mean sMAPE. The performance of all RNN-based models increases after combining them with traditional univariate forecasting models on the M3, M4, Ausgrid half-hourly, Ausgrid monthly, Tourism and Hospital datasets in terms of mean sMAPE. Furthermore, the ensemble model that combines the predictions of DTW.Number, ETS and ARIMA has provided the second-most accurate forecasts on the M4 monthly dataset based on the mean sMAPE, outperforming FFORMA. For the CIF 2016 dataset, forecast combinations show a mixed performance. Forecast combinations are substantially outperformed by the RNN-based models on the Kaggle web traffic dataset due to the poor performance of univariate

forecasting models. Overall, the performance of forecast combinations that use 2 univariate forecasting models are better than the combinations that only use a single univariate forecasting model.

In summary, the results indicate that the accuracy of globally trained models can be further increased by combining them with locally trained univariate models as they incorporate the strengths of both global and local models while mitigating the weaknesses of each other. These forecast combination models contain heterogeneous base models and hence, they are more capable of addressing the heterogeneity in the time series.

5.7. Computational Performance

Our experiments are extensive, which is why we have to execute them on different hardware and thus computational times are in general 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 (2.6GHz) and 32GB of main memory.

Table 6 shows the computational times of twenty models executed in the controlled environment for the Kaggle web traffic dataset. This dataset contains 997 series where each series contains 550 data points. It is the second largest dataset in terms of total data points. Hence, we consider the Kaggle web traffic dataset as a representative dataset to compare the computational times across models.

For each ensemble and non-ensemble clustering model of each baseline GFM, one model is run and its computational time recorded, as we assume that the time taken to cluster is small compared with the time taken to build the models, so that computational times are similar across the clustering variations of the same baseline GFM. The reported computational times of RNN and FFNN based models include hyperparameter tuning time, training time, and forecast calculation time.

We can see from the table that RNN and PR based models have the highest and lowest computational times, respectively. The ensemble of specialists has the highest computational time across RNN and PR based models whereas Kmeans.Number has the highest computational time across FFNN based models. Traditional univariate benchmarks, DeepAR, N-BEATS and FFNN based models have similar computational times which are higher than the computational time of PR models and lower than the computational time of RNNs. CatBoost and the PR Baseline have similar computational performance. Though the ensembling procedures approximately double the computational complexity, we note that the choice of base model is more important for the computational complexity, with RNNs being approximately 8 times slower than FFNNs, and FFNNs being about 20 times slower than the linear PR approach. Thus, we conclude that the choice of base model is more important than the choice for ensembling, regarding computation time, and ensembling is feasible for all base models considered.

5.8. Overall Performance of Ensemble and Non-Ensemble Models

Overall, the proposed ensemble variants outperform the non-ensemble models across all experimental datasets, except on the Kaggle web traffic dataset. The performance of the

Model	Computational Time (Minutes)
ETS	7
ARIMA	16
FFORMA	45
CatBoost	0.39
DeepAR	9
N-BEATS	21
RNN Baseline	68
RNN Kmeans.OC	65
RNN Kmeans.Number	109
RNN Ensemble.Specialists	114
RNN Ensemble.Seed	89
FFNN Baseline	9
FFNN Kmeans.OC	8
FFNN Kmeans.Number	24
FFNN Ensemble.Specialists	15
FFNN Ensemble.Seed	14
PR Baseline	0.38
PR Kmeans.OC	0.08
PR Kmeans.Number	0.45
PR Ensemble.Specialists	5

Table 6: Computational times (in minutes) of twenty models for Kaggle web traffic dataset. The benchmark models, the RNN-based models, the FFNN-based models, and the PR-based models are separated with double lines. One computational time is reported per each ensemble and non-ensemble clustering type of each baseline GFM as the computational times of the clustering variations with a given baseline GFM are similar.

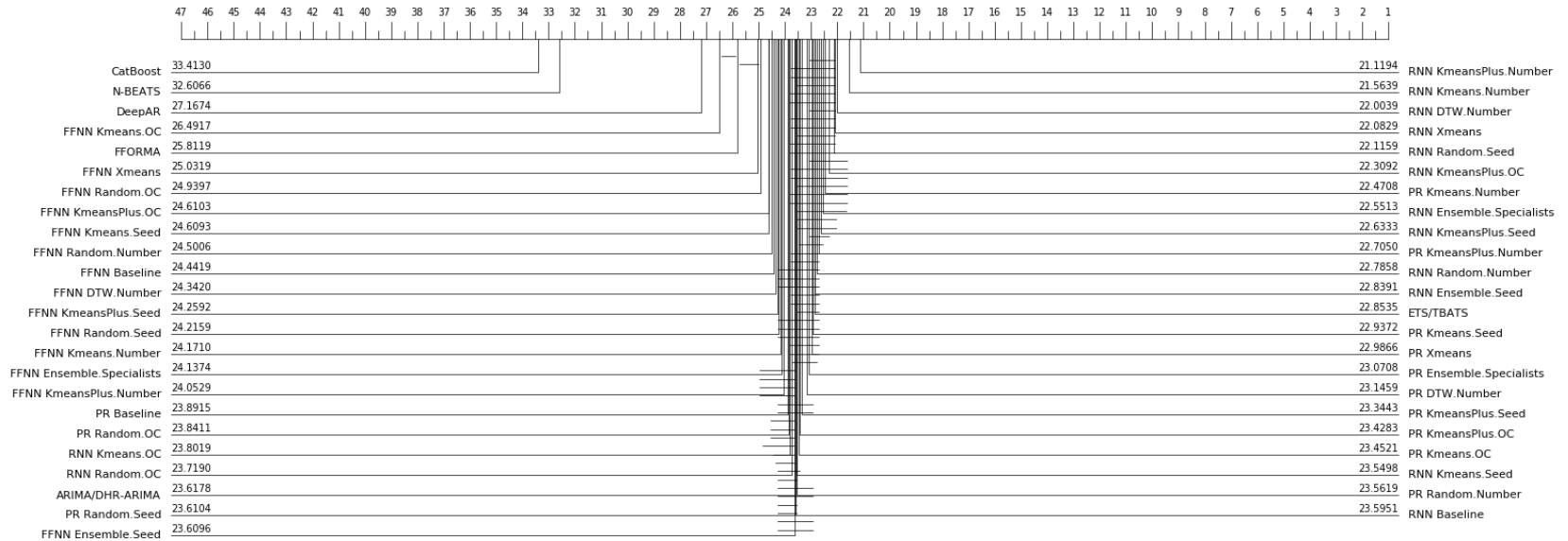

Figure 6: Critical difference diagram showing statistical difference comparison of all considered models. The models are ordered based on their average ranks, which are indicated next to the models, starting from the top right model. The horizontal lines that connect the groups of models indicate that the connected models are not significantly different from each other. Twelve GFMs show a significantly better performance than the traditional univariate benchmarks: ETS, ARIMA, TBATS and DHR-ARIMA.

ensemble variants depend on the characteristics of the datasets. The Kaggle web traffic dataset is a homogeneous dataset, where most of the time series share similar characteristics. For such datasets, global models that share the same parameters across all series perform best. The clustered ensemble models and the ensemble of specialists perform better on heterogeneous datasets as they group similar series and train different submodels per each subgroup. Hence, ensembling is quite helpful in increasing the forecasting accuracy, especially when the time series data are heterogeneous.

Figure 6 shows a critical difference diagram comparing the models we considered for our experiments. The average ranks are indicated next to the models. The horizontal lines that connect the groups of models indicate that the connected models are not significantly different from each other. Twelve GFMs show a significantly better performance than the traditional univariate benchmarks: ETS, ARIMA, TBATS and DHR-ARIMA. The KmeansPlus.Number cluster variant based on RNNs shows the overall best performance, whereas CatBoost shows the worst performance, out of all considered models. The FFNN-based GFMs overall show a poor performance compared to the RNN and PR-based GFMs. Our ensembled GFMs outperform the benchmarks, based on their average ranks.

As shown in Figure 6, models based on random clustering, namely Random.OC, Random.Number, and Random.Seed, show a better performance compared with the baseline GFMs across PR models. Across RNNs, the Random.Number and Random.Seed models show a better performance whereas across FFNNs, the Random.Seed model shows a better performance compared with the corresponding baseline GFMs. Hence, our results show that increasing the model complexity can improve the forecasting accuracy. When the series are partitioned with more sophisticated clustering techniques to train multiple GFMs, the model complexity gets further increased and the data heterogeneity issues are also properly addressed showing an improved performance over both baseline GFMs and random clustering based GFMs. Ensembling GFMs trained over multiple iterations further increases the model complexity, addresses data, model and parameter uncertainties, and mitigates the information loss that may occur due to the sub-optimal grouping of series. Hence, our clustered ensemble models obtain the top ranks in our evaluation.

Based on our results, we recommend to use GFM.Cluster.Number approaches over GFM.Cluster.Seed and Ensemble.Specialists approaches to address data heterogeneity (Section 5.4). We further recommend to use the distance-based clustering variant, DTW.Number, or the feature-based clustering variant, KmeansPlus.Number, as the base for GFM.Cluster.Number techniques (Section 5.1). As the base GFM, we recommend to use RNNs or PR models over FFNNs (Section 5.5).

6. Conclusions and Future Research

Global forecasting models that are trained across time series have recently become popular in the field of forecasting due to their better performance compared with univariate models that are trained per time series.

In this paper, we have investigated general methodologies to improve the accuracy of GFMs by better localising them, using clustered ensemble models, ensembles of specialists,

and ensembles of local and global models, to adequately address data heterogeneity and control model complexity. We have used three base GFMs, namely RNNs, FFNNs, and PR models.

Across eight publicly available datasets, we have shown that ensembling can improve the forecasting accuracy of GFMs trained over heterogeneous time series compared with the non-ensemble models such as GFM Baselines, clustered non-ensemble models, two deep learning benchmarks: DeepAR and N-BEATS, and four traditional univariate benchmarks: ETS, ARIMA, TBATS, and DHR-ARIMA, .

From our experiments, we conclude that the performance of GFMs can be improved by localising them and by properly addressing data heterogeneity during model training. We further conclude that ensembling is beneficial if a dataset contains diverse time series. Ensembling does not considerably improve the forecasting accuracy for homogeneous datasets, namely the Kaggle web traffic dataset in our experiments, where the GFM Baselines produce better forecasts. The forecast combinations that use both global and local model forecasts have provided the best results over many datasets as they incorporate the strengths of both global and local models while mitigating the weaknesses of each other.

Some areas for further research are as follows. In this research, we have only used simple averaging to aggregate the forecasts provided by the localised GFMs. Weighted averaging can be further incorporated to improve the performance of our proposed ensembled localised GFMs. Furthermore, we have only considered point forecasts in a univariate context. Ensembling can be further incorporated to address data uncertainty through probabilistic forecasting. Also, many real-world forecasting problems are multivariate, including more than one time-dependent variable and ensembling can be further useful to improve the forecasting accuracy of such forecasting problems.

Acknowledgements

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.

References

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

[2] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The M4 competition: results, findings, conclusion and way forward. *Int. J. Forecast.*, 34(4):802–808, 2018.

[3] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The M5 accuracy competition: results, findings and conclusions, 2020.

[4] R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science and Business Media, 2008.

[5] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley and Sons, 2015.

[6] P. Montero-Manso and R. J. Hyndman. Principles and algorithms for forecasting groups of time series: locality and globality. *International Journal of Forecasting*, 2021. ISSN 0169-2070.
[7] H. Hewamalage, C. Bergmeir, and K. Bandara. Global models for time series forecasting: a simulation study. https://arxiv.org/abs/2012.12485, 2020.
[8] J. R. Trapero, N. Kourentzes, and R. Fildes. On the identification of sales forecasting models in the presence of promotions. *Journal of the Operational Research Society*, 66(2):299 – 307, 2015.
[9] A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press, 2007. ISBN 9780521686891.
[10] S. Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.
[11] R. Sen, H-F Yu, and I. Dhillon. Think globally, act locally: a deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems*, pages 4837–4846, 2019.
[12] K. Bandara, C. Bergmeir, and S. Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: a clustering approach. *Expert Syst. Appl.*, 140:112896, 2020. ISSN 0957-4174.
[13] S. Lerch and S. Baran. Similarity-based semilocal estimation of post-processing models. *Royal Statistical Society*, 66(1):29–51, 2016.
[14] R. Godahewa, C. Bergmeir, G. I. Webb, and P. Montero-Manso. A strong baseline for weekly time series forecasting. https://arxiv.org/abs/2010.08158, 2020.
[15] R. E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, page 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
[16] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
[17] J. M. Bates and C. W. Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451—468, 1969.
[18] A. Timmermann. Forecast combinations. *Handbook of Economic Forecasting*, 1:135–196, 2006.
[19] D. H. Wolpert. The supervised learning no-free-lunch theorems. In R. Roy, M. Koppen, S. Ovaska, T. Furuhashi, and Hoffmann F., editors, *Soft Computing and Industry*, London, 2002. Springer.
[20] G. Brown, J. L. Wyatt, and P. Tino. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, page 1621–1650, 2005.
[21] S. P. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions*, 28(2): 129–137, 1982.
[22] D. Arthur and S. Vassilvitskii. K-means++: the advantages of careful seeding. In *Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms*, volume 8, pages 1027–1035, 01 2007.
[23] X. Jin and J. Han. *Encyclopedia of Machine Learning*. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8.
[24] G. T. Duncan, W. L. Gorr, and J. Szczypula. *Forecasting Analogous Time Series*, volume 30. Springer, Boston, MA, 2001.
[25] S. Smyl and K. Kuber. Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks. In *36th International Symposium on Forecasting*, 2016.
[26] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. DeepAR: probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.*, 36(3):1181–1191, July 2020.
[27] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. https://arxiv.org/abs/1905.10437, 2019.
[28] H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent neural networks for time series forecasting: current status and future directions. *International Journal of Forecasting*, 2020. ISSN 0169-2070.
[29] C. S. Bojer and J. P. Meldgaard. Kaggle forecasting competitions: an overlooked learning opportunity. *International Journal of Forecasting*, 2020. ISSN 0169-2070.
[30] D. M. Kent and R. A. Hayward. Limitations of applying summary results of clinical trials to individual patients - the need for risk stratification. *JAMA*, 298(10):1209–1212, 09 2007.

[31] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, page 226–231, 1996.

[32] L. Kaufman and P. J. Rousseeuw. *Partitioning Around Medoids (Program PAM)*, pages 68–125. John Wiley and Sons, Inc., 1990. ISBN 9780470316801.

[33] C. S. Wallace and D. L. Dowe. Intrinsic classification by MML-the snob program. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, volume 37, page 44, 1994.

[34] K. Bandara, P. Shi, C. Bergmeir, H. Hewamalage, Q. Tran, and B. Seaman. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *26th International Conference on Neural Information Processing*, pages 462–474, 2019.

[35] K. Bandara, C. Bergmeir, S. Campbell, D. Scott, and D. Lubman. Towards accurate predictions and causal 'what-if' analyses for planning and policy-making: a case study in emergency medical services demand. In *International Joint Conference on Neural Networks*, 2020.

[36] K. Bandara, H. Hewamalage, Y.-H. Liu, Y. Kang, and C. Bergmeir. Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognition*, page 108148, 2021. ISSN 0031-3203.

[37] V. R. R. Jose and R. L. Winkler. Simple robust averages of forecasts: some empirical results. *International Journal of Forecasting*, 24(1):163–169, 2008.

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

[39] A. Krogh and J. Vedelsby. Neural network ensembles, crossvalidation and active learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 231–238, 1995.

[40] G. Brown, J. L. Wyatt, R. Harris, and X. Yao. Diversity creation methods: a survey and categorisation. *Journal of Information Fusion*, 6(1):5–20, 2005.

[41] V. Cerqueira, L. Torgo, M. Oliveira, and B. Pfahringer. Dynamic and heterogeneous ensembles for time series forecasting. In *IEEE International Conference on Data Science and Advanced Analytics*, pages 242–251, 2017.

[42] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[43] L. Torgo and M. Oliveira. Ensembles for time series forecasting. In *Asian Conference on Machine Learning*, pages 360–370, 01 2014.

[44] J. Heinermann and O. Kramer. Machine learning ensembles for wind power prediction. In *Renewable Energy*, volume 89, pages 671–679, 2016.

[45] G. Grmanova, P. Laurinec, V. Rozinajova, A. Ezzeddine, M. Lucka, P. Lacko, P. VrablecovÃ¡, and P. Navrat. Incremental ensemble learning for electricity load forecasting. *Acta Polytechnica Hungarica*, 13:97–117, 2016.

[46] M. H. D. M. Ribeiro, V. H. A. Ribeiro, G. Reynoso-Meza, and L. d. S. Coelho. Multi-objective ensemble model for short-term price forecasting in corn price time series. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019.

[47] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[48] P. Laurinec, M. Loderer, M. Lucka, and V. Rozinajova. Density-based unsupervised ensemble learning methods for time series forecasting of aggregated or clustered electricity consumption. *Journal of Intelligent Information Systems*, 53(2):219–239, 2019.

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

[50] M. Pawlikowski and A. Chorowska. Weighted ensemble of statistical models. *International Journal of Forecasting*, 36(1):93 – 97, 2020. ISSN 0169-2070. M4 Competition.

[51] V. Cerqueira, L. Torgo, F. Pinto, and C. Soares. Arbitrage of forecasting experts. *Machine Learning*, 108(6):913–944, 2019.

[52] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley, 1990. ISBN 978-0-47031680-1.

[53] T. Warren Liao. Clustering of time series data — a survey. *Pattern Recognit.*, 38(11):1857–1874, 2005.
[54] X. Wang, K. Smith, and R. J. Hyndman. Characteristic-based clustering for time series data. *Data Min. Knowl. Discov.*, 13(3):335–364, 2006.
[55] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.
[56] V. Genre, G. Kenny, A. Meyler, and A. Timmermann. Combining expert forecasts: can anything beat the simple average? *International Journal of Forecasting*, 29(1):108–121, 2013.
[57] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, 1991.
[58] J. L. Elman. Finding structure in time. *Cogn. Sci.*, 14(2):179–211, 1990.
[59] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
[60] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002.
[61] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa. 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, 2012.
[62] A. M. Schafer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In *Artificial Neural Networks – ICANN*, pages 632–640. Springer Berlin Heidelberg, 2006.
[63] K. Bandara, C. Bergmeir, and H. Hewamalage. LSTM-MSNet: leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
[64] R. Godahewa, C. Deng, C. Bergmeir, and A. Prouzeau. Simulation and optimisation of air conditioning systems using machine learning. https://arxiv.org/abs/2006.15296, 2020.
[65] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.
[66] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.
[67] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.
[68] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. STL: a seasonal-trend decomposition procedure based on loess. *J. Off. Stat.*, 6(1):3–33, 1990.
[69] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2nd edition, 2018.
[70] R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software, Articles*, 27(3):1–22, 2008.
[71] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.
[72] G. M. Goerg. LICORS: light cone reconstruction of states - predictive state estimation from spatio-temporal data, 2013. R package version 0.2.0.
[73] A. Sardá-Espinosa. Time-series clustering in R using the dtwclust package. *The R Journal*, 2019.
[74] R. J. Hyndman, E. Wang, and N. Laptev. Large-scale unusual time series detection. In P. Cui, J. Dry, C. Aggarwal, Z. Zhou, A. Tuzhilin, H. Xiong, and X. Wu, editors, *IEEE International Conference on Data Mining Workshop (ICDMW)*, page 1616–1619, United States of America, 2015. IEEE, Institute of Electrical and Electronics Engineers.

[75] R. J. Hyndman, Y. Kang, P. Montero-Manso, T. Talagala, E. Wang, Y. Yang, and M. O'Hara-Wild. tsfeatures: time series feature extraction, 2020. R package version 1.0.2.9000.
[76] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones. catch22: canonical time-series characteristics. In *Data Min Knowl Disc*, page 1821–1852, 2019.
[77] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018. ISSN 0925-2312.
[78] A. Kassambara and F. Mundt. factoextra: extract and visualize the results of multivariate data analyses. R package version 1.0.5. https://cran.r-project.org/web/packages/factoextra, 2019.
[79] A. M. De Livera, R. J. Hyndman, and R. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *J. Am. Stat. Assoc.*, 106(496):1513–1527, 2011.
[80] R. J. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen. forecast: forecasting functions for time series and linear models, 2021. R package version 8.15.
[81] S. Makridakis and M. Hibon. The M3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000.
[82] M. Štěpnička and M. Burda. On the results and observations of the time series forecasting competition CIF 2016. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2017.
[83] Google. Web traffic time series forecasting, 2017. URL https://www.kaggle.com/c/web-traffic-time-series-forecasting.
[84] AusGrid. Solar home electricity data, 2019. URL https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data.
[85] G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011.
[86] R. J. Hyndman. expsmooth: data sets from forecasting with exponential smoothing. https://cran.r-project.org/web/packages/expsmooth, 2015.
[87] F. Orabona and T. Tommasi. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems*, volume 30, page 2160–2170, 2017.
[88] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011.
[89] M. Lindauer, K. Eggensperger, M. Feurer, S. Falkner, A. Biedenkapp, and F. Hutter. SMAC v3: algorithm configuration in python. https://github.com/automl/SMAC3, 2017.
[90] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *Int. J. Forecast.*, 22(4):679–688, 2006.
[91] A. Suilin. kaggle-web-traffic. https://github.com/Arturus/kaggle-web-traffic, 2017.
[92] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Turkmen, and Y. Wang. GluonTS: probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21 (116):1–6, 2020.
[93] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
[94] P. Montero-Manso. M4metalearning: metalearning tools for time series forecasting, 2020. R package version 0.0.0.9000.
[95] D. Pelleg and A. Moore. X-means: extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, page 727–734, 2000.
[96] K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232, 2009.

[97] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 01 2006.

[98] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[99] A. Benavoli, G. Corani, and F. Mangili. Should we really use post-hoc tests based on mean-ranks? *Journal of Machine Learning Research*, 17(1):152–161, 2016.

[100] H. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.