Erasmus University Rotterdam
Erasmus School of Economics

Master Thesis Econometrics and Management Science

Learning with Subset Stacking for Timeseries

By Stephen Tseng -450727

April 30, 2023

Supervisor: O'Neill, EP

Second assessor: Akyuz, MH

[1]The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

ABSTRACT

Forecasting the future by means of timeseries is a crucial task in any type of industry. There exists a wide range of forecasting techniques, but choosing the right one is deemed difficult. In our research we extend research in the forecasting literature by combining the topics of Machine Learning, Cross-Learning, Meta-Learning and Stacking to develop a new method for forecasting timeseries data. The new method we propose is Learning with Subset Stacking for Timeseries (LESST). The method is based on the Learning with Subset Stacking model by Birbil et al. (2021). We develop the model by means of the M4-competition dataset. The method considers clustering subsets of timeseries using K-Means clustering. For the timeseries subsets, Local Models are trained which are then weighted and combined in a Global Model. As a benchmark model we make use of the Theta model that won the M3-competition. With LESST we are able to beat the Theta model solely for one out of the six datasets of the M4-competition. On some occasions LESST performs closely with the Theta model. In conclusion, the LESST model performs competitively, however, it is not yet a state of the art method. There is more room for improvement by means of further research.

Contents

1 Introduction

The success of organizations relies on good decision making and planning. To make these decisions wisely and plan well, is a challenging task, as the future holds many uncertainties. As organizations seek to minimize their risk and maximize their utilities, forecasting has been an important tool for good decision making and planning. In the real world there are a lot of applications for forecasting, which calls for a diverse range of forecasting methods to tackle real-life challenges. In essence there are a lot of different models to consider for producing forecasts. These models all rely on time series data of good quality, as we would like to develop good quality forecasts for future values.

Essentially, there are two approaches to forecasting time series nowadays: generating forecasts from a single model and combining forecasts from a set of different models. Combining different models into one forecast is called forecast model averaging. A lot of research has already been developed on the combination of different forecasting models, of which many empirical applications show that the combination of forecasting models is often superior to that of an individual model (Yin et al., 2012). Combining forecast models by using a weighted average is a simple and effective method, as it avoids the risk of selecting a mis-specified model. A major challenge, however, is to select a set of weights that optimizes the accuracy of the model. Often the resulting weights cause a worse model performance than when equal weights are used (Wang et al., 2022).

As of late, Artificial Intelligence (AI) has become popular and has frequently been used in practice and research. However, as the methods are still quite young it still requires a lot of research as compared to statistical methods. Machine Learning (ML), which is a branch of AI, is often used in Forecasting context as of late. However, research has shown that classical statistical methods often outperform ML methods in terms of forecasting (Makridakis et al., 2018).

The literature on ML methods in forecasting often do not use proper benchmarks to compare their methods, resulting in untrustworthy results. Even though, ML has some drawbacks

in forecasting, it shouldn't be dismissed. Further research is required, which is also why the M-competitions are organized. These competitions bring researchers together and have them compare their methods in terms of forecasting accuracy. In the M3 competition the winner consisted of a statistical method, the Theta model (Assimakopoulos & Nikolopoulos, 2000).

Remarkably, the method that made first place in the most recent competition (M4) was not purely a statistical method but instead, a combination of ML and a traditional statistical method (Smyl, 2019). Cross-Learning (CL) methods were among the top models in the competition. The Cross-Learning methods leverage the information across all the different time series to build their forecasting models. However, not all Cross-Learning methods performed well, therefore Semenoglou et al. (2020) investigated these methods further. Their findings found that simple cross-learning methods easily outperformed traditional forecasting methods and are even further enhanced by making use of time series features. The accuracy of these methods however, come at a great computational cost as compared to traditional statistical methods. Clearly, not all CL methods can be covered within one paper, hence further research is needed to decide which CL approaches are more suitable for what instance.

Therefore, we aim to extend research in the forecasting literature by combining the topics of Machine Learning, Cross-Learning, Meta-Learning and Stacking to develop a new method for forecasting timeseries data. To achieve this, we make use the M4 data competition set. Our method is based on the idea of Learning with Subset Stacking (LESS), which was developed by Birbil et al. (2021). LESS is a method which leverages the heterogeneous behaviour between the input and output space. The method consists of a meta-learning process in which the input is clustered into multiple sub-samples. For these sub-samples, multiple Local Models are trained and then combined using a form of stacking. In their method they weight the Local Models by the distance to other subsets, which are in the end combined into one Global Model. LESS has been tested on multiple datasets and has shown to perform competitively.

However, the method has not been developed for forecasting timeseries. Our method will consider the same framework, in which we will train Local Models for the different subsets of

the time series data. Finally, we develop a global model that combines the results from the multiple Local Models. Instead of creating subsets that contain data points as considered in Birbil et al. (2021), we will create subsets that contain multiple timeseries. For the clustering approach we have decided to use a feature based clustering approach. This is an approach that uses time series features of the timeseries to compile a K-means clustering method. Subsequently, it will allocate our series in different sets that share similar timeseries properties.

In the Local Models, Birbil et al. (2021) considers the use of the Ordinary Least Square (OLS) method. With our model we have the capability of using OLS as well. However, we use the model with a direct pooled forecasting approach using the different time series. In addition, we also consider the use of other methods for forecasting. The first methods being tree-based methods, where we make use of the RandomForest, XGBoost and LGBM method and the latter being a more robust approach to OLS, namely the Huber regression.

Lastly, we combine these local models into one global model. For the Global Model we are able to use the same type of methods as we do in the local model. However, similarly to Birbil et al. (2021), the input to the Global Model will be the weighted forecasts of the Local Models.

As a benchmark for our method, we will consider using Theta model that was proposed by Assimakopoulos & Nikolopoulos (2000). This model received the first place in the M3 competition. The method considers deseasonalization of the data if applicable and makes use of classical statistical methods. It is therefore commonly used as a benchmark, especially for methods developed for the M4 competition.

2 Literature review

In this section we will discuss the relevant literature on forecasting models. Birbil et al. (2021) proposes a new learning algorithm for a set of input-output pairs. Their method Learning with Subset Stacking (LESS), considers a three step process for developing a model. These steps consist of clustering data points together into subsets, training local models for each

subset and developing a global model from a combination of weighted local model predictions. In their research they obtain competitive results with other state-of-the-art methods. Their method consists of stacking models together from separate clusters, however, it does not consider the use of time series data. In our research we aim to extend the concepts used in this method for use of timeseries forecasting.

Combining machine learning and standard forecasting methods has become more common. Zhao & Feng (2020) develop a framework which takes forecasts developed by the standard methods as inputs into a machine learning model. Their intuition regarding the inputs is that the forecasts are good features that characterise the time series well. The machine learning models they propose are the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN) model. Their method For2For outperforms all M4 data competition submissions for quarterly data and performs highly for monthly series. However, their method suffers from over-training for small sample sizes. Another point they note is that using a single RNN model for all frequencies performs better overall than separate models. The model considered in this research for combining the forecasts could prove beneficial in the Global Model of our method. However, as we do not consider the use of Neural Network models in our research, we leave it for future reference.

The model that reached second place in the M4 data competition also considered stacking methods in combination with cross-learning. This method uses time series features to build a meta-model that calculates weights for combining different forecasting models. It has been developed by Montero-Manso et al. (2019) and outperforms simple forecasting combination models as well as most methods in the time series forecasting literature. By combining time series features for each individual time series within a training set, the model is able to train a meta-model that produces weights. Their meta-model is trained by use of the xgboost optimizer. For any new series to be forecasted, the features are required and given to the meta-model to produce weights for combining forecasts. They claim their method to be robust as removing a model does not alter the forecasts strongly. The idea of using timeseries features for developing a model to produce weights is interesting. The weights we develop in our method for combining Local Models are dependent on the timeseries features. Although, these weights are created by means of distances from clusters. It could be interesting to use

the features themselves for developing weights as done in Montero-Manso et al. (2019) for producing weights to combine the Local Models.

The importance for deseasonalizing the data of the M4 competition is heavily emphasised in Smyl (2019). Hence, we consider to incorporate deseasonalization as a part of our method. His method won first place in the M4 competition. It considers a hybrid method for forecasting by combining exponential smoothing and Recurrent Neural Networks. The method is hybrid in the sense that it combines traditional statistical methods with machine learning, in this case a recurrent neural network. The first step in the framework consists of performing exponential smoothing. This is done before the data is used for training a Long Short-Term Memory model (LSTM), the output of the LSTM model is then re-transformed. With this method, he combines global and local parameters to enable cross-learning and giving the method a hierarchical nature. The use of exponential smoothing might be a good additional component to our method for future reference. Especially when Neural Network models are being used, since it has proved beneficial for enhancing performance (Smyl, 2019).

Godahewa et al. (2021) investigate the improvement of Global Forecasting Models (GFM). They localize the GFM by using clustered ensemble models. Similarly, their method considers feature-based clustering and distance clustering using Dynamic Time Warping (DTW). However, their method is inherently different from what we are considering. Their framework considers training the Global Model iteratively per cluster, producing forecasts in each iteration. In the end all the forecasts for each iteration are averaged over and used as final forecasts. Using their localization method they successfully conclude that it improves the GFM. This framework could promise interesting for combining our Local Model results. Instead of one Global Model prediction there would be multiple predictions that are aggregated. This could be either done by using the model as described in Godahewa et al. (2021) or by using multiple global models (e.g. Random Forest) that would then be trained iteratively with their forecasts being aggregated.

In Semenoglou et al. (2020) the added benefit of cross-learning is investigated. They consider a wide variety of different cross-leaning methods. As their base model they use a Neural Network model. Other varieties include feature-based methods and clustering. Likewise, they consider feature-based clustering for separating the dataset into subsets. However,

they only consider training local models per cluster, without leveraging information from the other clusters. They conclude that the utilization of cross-learning improves forecasting accuracy. Furthermore, the method leveraging feature based methods outperformed all of their other methods, specifically the method using features as input to the model.

Liang et al. (2021) considers clustering timeseries of stocks using DTW clustering. They conclude that with the use of this method, they obtain more accurate clusters compared to many other traditional similarity methods. Even though this cluster method seems promising, we do not consider to use it in our method. This is mainly due to the running time of the clustering method. As we are considering a large dataset of many series, it would become infeasible to use this clustering method.

Grinsztajn et al. (2022) show the difference in performance of tree-based and deep learning methods for tabular data. They reach the conclusion that tree-based methods clearly outperform their deep learning counterparts, especially for datasets with less observations. Furthermore, the tree-based methods also require a smaller time frame to fit the data. Hence we have decided that it is more useful to focus on tree-based methods as opposed to a deep-learning method in our research.

3 Data

In this research we use the M4 Dataset from the M4 data competition. In this competition multiple researchers competed to produce the most accurate model using this specific dataset. It is the same dataset used in Montero-Manso et al. (2019) and Smyl (2017) to produce their models. The M4 dataset consists of 100,000 distinct timeseries which were all used in the Makridakis forecasting Competition. The timeseries data consists of multiple timeseries frequencies, namely: yearly, quarterly, monthly, weekly, daily and hourly. Furthermore, the data consists of timeseries data from different sectors. For each timeseries it is given as a covariate to which sector it belongs. The sectors present in the dataset are macro, micro, industry, financial, demographic and other data. For each time frequency, the series are split

into a train and test set. Hence, we have 6 different sets of training data and 6 different sets of testing data. What is important to note is that not all timeseries within the training dataset consist of the same number of observations. A brief summary of the number of timeseries can be viewed in table 1. The M4 dataset we obtain from Kaggle (2020).

Table 1: Overview M4 dataset

Frequency	**Micro**	**Industry**	**Macro**	**Finance**	**Demographic**	**Other**	**Total**
Yearly	6,538	3,716	3,903	6,519	1,088	1,236	**23,000**
Quarterly	6,020	4,637	5,315	5,305	1,858	865	**24,000**
Monthly	10,975	10,017	10,016	10,987	5,728	277	**48,000**
Weekly	112	6	41	164	24	12	**359**
Daily	1,476	422	127	1,559	10	633	**4,227**
Hourly	0	0	0	0	0	414	**414**
Total	**25,121**	**18,798**	**19,402**	**24,534**	**8,708**	**3,437**	**100,000**

Source: Makridakis et al. (2020)

In our method we do not separate the data for different sectors, hence we do not make use of any covariates. Therefore, we are left with the amount of timeseries in the Total column that can be viewed in table 1. We define the dataset of timeseries as Y, we develop models for each frequency separate but in the exact same manner. The dataset Y consists of timeseries y_i, where $i = 1, 2, ..., N$ and N is the number in the **Total** column in table 1 depending on the dataset we are using. Furthermore, the dataset Y has already been split into a training set Y_{train} and a test set Y_{test} by the organizers of the M4 competition. Y_{test} has the dimension of N by h, where h is the amount of steps ahead that need to be forecasted by the model.

Table 2: Details M4 dataset

	Prediction steps	**Avg. data points**	**Total data points**
Yearly	6	31	720,458
Quarterly	8	92	2,214,108
Monthly	18	216	10,382,411
Weekly	13	1022	366,912
Daily	14	2357	9,964,658
Hourly	48	854	353,500

Each dataset considers a different amount of steps to be forecasted, for each dataset the amount for h can be found in table 2 in the column prediction steps. Furthermore, in table 2 we include the total amount of data points in the training set of all timeseries combined and the average amount of data points for each timeseries for each dataset. However, each series within the dataset has a different time frame (length), hence we denote the observation at time t as t_i for series i. The dimension of the training set Y_{train} are therefore N by t_i. In the methodology 4 we refer to Y_{train} simply as Y for simplicity, where Y is defined as in equation 1.

$$Y = \{y_i = y_{i,1}, y_{i,2}, ..., y_{i,t_i} | i = 1, ..., N\} \tag{1}$$

For any series y_i in the dataset it is possible that it has seasonal patterns or is not stationary. We tackle this in the section 4.1.6.

4 Methods

4.1 Learning with Subset Stacking for Timeseries

We consider adopting the framework of Learning with Subset Stacking (Birbil et al., 2021) and adjust it in such a way that it will leverage multiple time series for producing forecasts. The original method LESS clusters data points within a single series of data. Our method on the other hand considers the clustering of similar time series. We use the M4 dataset described in the data section 3 and consider a different model for each frequency.

Consider our dataset for an arbitrary frequency $Y = \{y_1, y_2, ..., y_N\}$ where N is the number of different time series. For each timeseries we calculate the Timeseries Features, we perform this process separately as it saves time for training multiple models. The Timeseries Features are used in the clustering step, where timeseries will be allocated in different subsets $\{Y^{(1)}, Y^{(2)}, ..., Y^{(K)}\}$ where K is the number of clusters considered. During the clustering process we also calculate a set of weights $W_{N\text{x}K}$, which is a N by K matrix where N is the number of series and K the number of clusters. The weights represent the odds (according to distance) for a series n to belong to a specific cluster k.

For each subset $Y^{(k)}$ where $k = \{1, 2, ..., K\}$ we train a different Local Model $\mathcal{L}(y|Y^{(k)})$. Once the subsets are formed between all series and the Local Models have been developed, we combine their forecasts by weighting the output of the Local Models using the weights produced in the clustering step, these are then saved into a matrix $Z = W\mathcal{L}(y|Y^{(1)}, Y^{(2)}, ..., Y^{(K)})$ (see section 4.1.14 for more details) which is a N by h matrix, where N is the number of series and h the forecasting horizon.

Finally, we use the produced Z matrix as input for our Global Model $\mathcal{G}(y|Z)$, which combines the different weighted local forecasts into one final forecast. We develop the model using the programming language python. In addition we make use of some external python libraries which can be viewed in the appendix 7.3.1

In section 4.1.1 we start off by explaining the process of obtaining the Timeseries Features. Secondly, we discuss the clustering method and the calculation of the weights in

Section 4.1.2. Followingly, we explain the process of deseasonalization of the data in Section 4.1.6. In section 4.1.4 we dive into further detail on the Local Models and lastly, in section 4.1.14 we finalize the details on the global model.

4.1.1 Timeseries Features

Before building the model we start off by processing the data. For each timeseries y in the training set Y we calculate a set of 41 different timeseries features. We are using the same features that have been used in the FFORMA model and are available in the R **forecasting** package. The features we consider with their respective explanation can be found in the appendix 7.2. The calculation is done for N different time series and results in $S \leq 41$ features. Features that were not able to be calculated for a series are removed from the set of features. We are left with a matrix $F_{N\text{x}S}$ (a N by S matrix), which contains all the features for the series in our set Y. This process is performed separately for each frequency of the data. As the M4 dataset consists of 6 different datasets with different time frequencies, we are left with 6 different datasets of timeseries features. We save each set for later use, as we will use these for the clustering step.

4.1.2 Timeseries Clustering

In this step we start dividing the different series in our training set Y into different subsets using the KMeans clustering algorithm. We make use of the previously calculated features in F where $N = |F|$ to cluster the series into K different clusters; $F_1, F_2, ..., F_K$. Our set F contains S different features that can be found in the appendix 7.2. Furthermore, F consists of $f_1, f_2, ..., f_N$ points in the space $\mathbb{R}^S$. The KMeans algorithm allocates each point $f_i,\ i = 1, 2, ..., N$ to a cluster with a center $c_k,\ k = 1, 2, ..., K$ where K is the number of clusters we consider. For our model we will consider several different cluster sizes, namely $K \in \{3, 10, 30, 50, 100\}$. The KMeans algorithm allocates the points to the clusters in such a way that it minimizes the k-means cost function $\sum_i^k \sum_{f \in F_i} \|f_i - c_i\|^2$. Minimizing the cost function is the key to minimizing the distance for each point to the cluster centers that are created by taking the center of the points within the cluster. Once the clustering algorithm reaches it's optimum, we save all cluster centers c_k into a matrix $C_{K\text{x}S}$ where K is the number of clusters and S the number of features. Furthermore, we save the indices i that belong to

cluster k into a mapping $I = \{k : i\}$. Using these cluster allocations, we create our subsets $Y^{(1)}, Y^{(2)}, ..., Y^{(K)}$. These subsets we use later on to train our Local Models.

4.1.3 Distance Weighting

In the LESS model the Local Model forecasts are weighted, these weights are created from the distance of a point to each cluster. We consider a similar process where we use the distance of the timeseries features of a series to the cluster. Calculating these distances is done by the KDTree method (Bentley, 1975) using the cluster centers C and the set of timeseries features F. We define the distances as $d_i = (d_{i1}, d_{i2}, ..., d_{iK})$, here $i = 1, 2, ..., N$ is the series, K is the number of clusters and d_{ik} the distance of series i to cluster k. The weights are calculated as defined in equation 2.

$$w_{ij} = \frac{\frac{\sum_{k=1}^{K} d_{ik}}{d_{ij}}}{\sum_{q=1}^{K} \frac{\sum_{k=1}^{K} d_{ik}}{d_{iq}}} = \frac{\frac{1}{d_{ij}}}{\sum_{q=1}^{K} \frac{1}{d_{iq}}} \tag{2}$$

Calculating this for all clusters we are left with the vector of weights $w_i = (w_{i1}, w_{i2}, ..., w_{iK})$ where $i = 1, 2, ..., N$. Combining these weights for all series i, we obtain the matrix of all weights $W_{N\text{x}K}$.

4.1.4 Local Model

During the clustering step we split our timeseries into K different subsets $Y^{(1)}$,$Y^{(2)}$,...,$Y^{(K)}$. For each of these subsets $Y^{(k)}$ we train a Local Model $\mathcal{L}(y_i|Y^{(k)})$ where $i = \{i = 1, 2, ..., N|y_i \in Y^{(k)}\}$ and $k = \{1, 2, ..., K\}$. The first step in training the Local Models will be to fit each time series y_i in $Y^{(K)}$ for a specific model. For this instance, we make use of a pooled direct forecasting approach, as we pool the different series in the subset K to fit one model. We consider several different regression methods that can be used interchangeably for the Local and Global Models. The chosen methods are the **Extreme Gradient Boosting**, **Light Gradient Boosting Machine**, **Random Forest**, **Linear Regression** and the **Huber Regression**. We have chosen several machine learning methods due to their performance and convenience of use for the structure of our model. Furthermore, we consider the simple linear regression due to its low computational cost and to compare to the more advanced machine learning methods. Lastly, the **Huber Regression** was chosen due

to its low computational cost and robustness towards outliers. We will not consider the **Random Forest** as a Local Model due to its running time, we instead use it in the Global Model step. Before any of these models can be fitted on our time series we have to prepare the data for training.

4.1.5 Data preparation for Local Models

First and foremost, before fitting our models we have to decide the amount of steps h that we will forecast ahead. The h ahead forecast we consider depends on the frequency of the dataset, as each frequency has its own test dataset with h data points for all the N timeseries. Let us consider our training set $Y^{(k)}$ that contains several series y_i with length t_i. For each series i in the cluster k we create an input and target set. Before we do this, we first omit the last h data points from the series i as these will be used later on in the Global Model and to prevent over-fitting. The model does include the functionality to include all points, however, we do not consider using this in our results. Once this is done we are left with the series $y_{i,1}, y_{i,2}, ..., y_{i,t_i-h}$, which we then split into the input vector $X^{(k)}_{i,input}$ and the target vector $X^{(k)}_{i,target}$. Since we are predicting h steps ahead using a direct pooled forecasting approach, we require to fit h different models. To form the input set we lose h data points as we require the same dimension of vectors to fit the h models simultaneously. Hence, we define the input for Local Model k and series i belonging to subset $Y^{(k)}$ as in equation 3 for each step to be predicted.

$$X^{(k)}_{i,input} = (y_{i,1}, y_{i,2}, ..., y_{i,t_i-2h}) \tag{3}$$

Here $i = \{i = 1, 2, ..., N | y_i \in Y^{(k)}\}$, meaning that for model k only series i is used in the input. To create the target vector $X^{(k)}_{i,target}$ is a bit more complicated. It must have the same length as the input vector $X^{(k)}_{i,input}$ of $t_i - 2h$. The difference is that the entries can vary between 1 and h steps ahead. The input for fitting the 1-step ahead model consists of $\{y_{i,2}, y_{i,3}, ..., y_{i,t_i-2h+1}\}$ while for the h-step ahead it consists of $\{y_{i,1+h}, y_{i,2+h}, ..., y_{i,t_i-h}\}$. Thus we define the target vector for $q = 1, 2, ..., h$ steps ahead as in equation 4.

$$X^{(k)}_{i,target} = (y_{i,1+q}, y_{i,2+q}, ..., y_{i,t_i-2h+q}) \tag{4}$$

Where $i = \{i = 1, 2, ..., N | y_i \in Y^{(k)}\}$ is the index of the timeseries, $k = 1, 2, ..., K$ is the cluster number and h the forecasting horizon.

4.1.6 Deseasonalization

Before we fit our Local Models, we consider deseasonalizing the data. For each series y_i we test whether the series shows seasonal behaviour that is significantly present. This test is performed using the lag auto correlation function r_k, that is the function for for the lag k. In equation 5 we show the test performed, which has been proposed in Fiorucci et al. (2016).

$$|r_m| > q_{1-a/2}\sqrt{\frac{1 + 2\sum_{i=1}^{m-1} r_i^2}{n}} \tag{5}$$

Here m is the frequency used for the data, for the monthly series it would be 12 for example, which is the seasonal cycle. Furthermore, the sample size is defined by n and q is the quantile function of the standard normal distribution. Lastly, $(1-a)\%$ is the confidence level for which we opt for 90%. See Fiorucci et al. (2016) for more details. Following the test, for each series y_i that is seasonal according to the test, we deseasonalize the data using a classical decomposition method. To reseasonalize the data after forecasting we need to use the seasonal components calculated during the deseasonalization step. We save the seasonal components for each series i in a set S during the deseasonalization step. Here S consists of $s_1, s_2, ..., s_N$, using s_i we can reseasonalize y_i after the deseasonalization was performed, where $i = 1, 2, ..., N$. The reseasonalization is performed on the Global Model predictions, resulting in the predictions for the LESST model using deseasonalization. As the predictions are only reseasonalized for the Global Model, the Local Models predict the deseasonalized series, for which the predictions go straight into the Global Model without reseasonalization.

4.1.7 Fitting the Local Models

With our input and target sets defined for each cluster k, we can now start fitting the local models. Consider our local model $\mathcal{L}(y|Y^{(k)})$, we fit our model in such a way that $X_{target}^{(k)} = \mathcal{L}(X_{input}^{(k)}) + \epsilon$, where ϵ is an error matrix. We perform this for $k = 1, 2, ..., K$ and save the models inside a mapping $L = \{1 : \mathcal{L}(y|Y^{(1)}), 2 : \mathcal{L}(y|Y^{(2)}), ..., K : \mathcal{L}(y|Y^{(K)})\}$ for later use. Due to the direct forecasting approach for h steps, we use h different models for

each cluster k. These different models are combined into one so that the **fit** method can be used in one instance. We will now discuss the different local models in more detail.

4.1.8 Linear Regression

Linear Regression is a standard regression model that uses the fundamentals of Ordinary Least Squares. To decide the value of the unknown parameters the method minimizes the sum of squared errors. A linear regression is defined in the following way:

$$y_i = \hat{\alpha} + x_i\hat{\beta} + e_i \tag{6}$$

Here $i = 1, 2, ..., N$, $\hat{\alpha}$ is the constant, $\hat{\beta}$ the parameter of interest, which shows the effect of the covariates x_i and e is the residual. In our case the covariate x_i consists of the lagged values of y_i. The sum of squared residuals is defined as $\sum_{i=1}^{N}(y_i - \hat{\alpha} - x_i\hat{\beta})^2$, which is then minimized to determine the values of the parameters. In practice, for OLS to produce the best estimates it is required for a set of assumptions to hold. However, in our research we are only interested in the predictive ability. In LESST the linear regression is mainly used as it is a simple method that is fitted in little time on a lot of data.

4.1.9 Huber Regression

The Huber Regression is a linear regression method that is more robust to outliers than OLS. The main difference between Huber and OLS is that it uses a different function to minimize over. Instead of minimizing the sum of squared errors it minimizes the following function.

$$min_\beta \sum_{i=1}^{N} \phi(y_i - \alpha - x_i\beta) \tag{7}$$

where ϕ is defined by

$$\phi(u) = \begin{cases} u^2, & \text{if } |u| \leq M \\ 2M|u| - M^2 & \text{if } |u| > M \end{cases} \tag{8}$$

Here M is some threshold value larger than zero. Even though it is very similar to OLS, it specifically treats large residuals in a different matter. As the sum of squared errors penalizes large residuals quadratically, the Huber loss function does this in a linear matter. This causes

the method to be affected less by large outliers in the data.

4.1.10 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a tree-based method that makes use of the machine learning technique tree-boosting. The method has been proposed by Chen & Guestrin (2016). It specifically makes use of the algorithm gradient-tree boosting which was introduced in Friedman (2001). The method consists of making an ensemble of different predictive tree-based models. The objective function which is optimized minimizes iteratively in the direction of the gradient. The method XGBoost is an upgrade from a standard gradient boosting algorithm as it converges more quick. This is due to the loss function being expanded with second order Taylors-expansion. XGBoost optimizes the following objective function (Chen & Guestrin, 2016):

$$\mathcal{L} = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{9}$$

Here $\hat{y}_i$ is the value estimated by the model and y_i is the actual value. Furthermore, l is a differentiable loss function, Ω a penalty function and k the number of classification regression trees. The decision tree model is defined by f, where f_k is the model belonging to tree k. The penalty function takes the tree model as input $\Omega(f)$ and is defined in the following matter:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\omega\|^2 \tag{10}$$

In this equation the penalty coefficients are defined by γ and λ. These coefficients prevent the model from becoming overly complex. Moreover, T is the number of leaves in the tree model and ω the sum of all the leaf weights

4.1.11 Light Gradient Boosting Machine

The Light Gradient Boosting Machine (LGBM) is just like XGBoost a gradient tree-boosting method. The method was first introduced in Ke et al. (2017). The fundamental difference between these two methods is that the tree-models in LGBM grow by their leaves and in XGBoost they grow in depth. Due to this difference it is possible for LGBM to outperform XGB but can also lead to the method to over-fit on the data (Ma et al., 2018). Overall, due to the fundamental difference in tree-growth, the LGBM method is computationally less

costly.

4.1.12 Random Forest

The Random Forest is a tree-model method that was first introduced in Breiman (2001). It is a machine learning method that combines multiple tree models together. Using the dataset given to the algorithm, it samples at random from the dataset for further use. Furthermore, the method defines several features on which the trees will be split into branches. However, instead of using all features, each tree-model will only leverage a random sub sample of all features. The method uses both ensemble and bagging techniques. Combining results from multiple models is considered an ensemble method. Moreover, bagging is the training of models on a random subset of the full dataset. Due to this process, the Random Forest method is less affected by a noisy dataset. For more in-depth details on the method, see the original paper (Breiman, 2001) or the in-depth research on the topic (Louppe, 2014).

4.1.13 Local Model Predictions

To evaluate the Global Model performance we consider to investigate the Local Model performance for the best performing LESST models. In the Global Model we use all Local Model predictions using all the series for each model, more details on this can be read upon in section 4.1.14. To evaluate the Local Model performance we develop forecasts for each series that belongs to the cluster of the Local Model. Therefore, for $y_i \in Y^{(k)}$ we make forecasts using the Local Model $\mathcal{L}(y_{t_i+q}|Y^{(k)})$, here $i = 1, 2, ..., N$ is the number of the series considered, $q = 1, 2, ..., h$ is the step ahead, $k = 1, 2, ..., K$ the number of the cluster and $Y^{(k)}$ the subset which the Local Model $\mathcal{L}(.|Y^{(k)})$ belongs to.

4.1.14 Global Model

With the Local Models trained, we are now ready to combine them into our global model. The first step now is to split our whole dataset into the input and target vectors for fitting the model. We consider two different methods of forming the input and target. Firstly, we cover the method that considers the use of a smaller amount of data points, which is the method we will use in our main results (section 5.1). For this method, we form a train set by removing h data points for each series, these h points are used in the target set. We are then left with the set X_{train} which has N by $t_i - h$ data points. Our target set we create by

taking those last h points for each series and saving them into a set X_{target}, a set with N by h data points, see equation 13. Followingly, we only use the last data point in the train set $(t = t_i - h)$ for each series i. These points are used to forecast the $q = 1, 2, ..., h$ different steps ahead for each series i using all the Local Models $\mathcal{L}(y|Y^{(1)}), \mathcal{L}(y|Y^{(2)}), ..., \mathcal{L}(y|Y^{(K)})$. Using the Local Model mapping L and the weight matrix W, we are able to create the weighted $q = 1, 2, ..., h$-step forecast. This step is performed for each series in the following way:

$$\{w_{i,1}\mathcal{L}(\hat{y}_{i,t_i-h+q}|Y^{(1)}), w_{i,2}\mathcal{L}(\hat{y}_{i,t_i-h+q}|Y^{(2)}), ..., w_{i,K}\mathcal{L}(\hat{y}_{i,t_i-h+q}|Y^{(K)})\} \tag{11}$$

Where we define $w_{i,k}$ as the weight for the series i and the Local Model belonging to cluster k. Furthermore, $\mathcal{L}(y_{t_i-h+q}|Y^{(k)})$ is defined as the $q = 1, 2, ..., h$-step ahead forecast for $y_{i,t=t_i-h}$ for the series i using the Local Model belonging to cluster k. Using equation 11 we define our input vector for series i and q-steps ahead as in equation 12. Where $i = 1, 2, ..., N$ is the series and $q = 1, 2, ..., h$ the q-step ahead true values.

$$X_{i,input} = (w_{i,1}\mathcal{L}(\hat{y}_{i,t_i-h+q}|Y^{(1)}), w_{i,2}\mathcal{L}(\hat{y}_{i,t_i-h+q}|Y^{(2)}), ..., w_{i,K}\mathcal{L}(\hat{y}_{i,t_i-h+q}|Y^{(K)}) \tag{12}$$

Combining for each series i and step ahead q we obtain the set X_{input} that consists of $N \cdot h \cdot K$ data points. Before we train our Global Model, we define the target $X_{i,target}$ of the target set X_{target} in equation 13

$$X_{i,target} = y_{i,t_i-h+q} \tag{13}$$

Using our input and target defined as in equation 12 and 13 we are capable of training the Global Model. For training the model we can use any of the model types which we have mentioned previously in section 4.1.4.

The second method we consider for forming the target and input set considers significantly more data points for the Global Model. First of all, instead of only predicting the h different steps ahead for the last data point at $t = t_i - h$ in X_{train}, we use all data points from 1 to $t_i - h$. Subsequently, we end up with $\sum_{i=1}^{N}(t_i - h) \cdot h \cdot K$ Local Model predictions.

Looking back at table 2 makes it clear that this considers a substantial larger amount of data points going into the Global Model. Therefore, we do not consider using this method in the main results but instead use it to evaluate the Global Model performance in section 5.3. Following this process, we end up with the input vector as defined in equation 14. Where $t = 1, 2, ..., t_i$ is the time point of series $i = 1, 2, ..., N$ for the $q = 1, 2, ..., h$-step ahead being predicted.

$$X_{i,input} = (w_{i,1}\mathcal{L}(\hat{y}_{i,t-h+q}|Y^{(1)}), w_{i,2}\mathcal{L}(\hat{y}_{i,t-h+q}|Y^{(2)}), ..., w_{i,K}\mathcal{L}(\hat{y}_{i,t-h+q}|Y^{(K)}) \tag{14}$$

Comparing equation 12 with 14 we see that they are exactly the same for $t = t_i$. Lastly, we define the target for the input in equation 15.

$$X_{i,target} = y_{i,t-h+q} \tag{15}$$

where $t = 1, 2, ..., t_i$ is the time point, $i = 1, 2, ..., N$ the series, $q = 1, 2, ..., h$ the time step of the true value and h the forecast horizon.

The whole process of training and making forecasts using the LESST algorithm is summarized using psuedocode in the algorithm 1.

Algorithm 1: How to train and forecast using the LESST model

Result: LESST model and predictions
set number_clusters $\leftarrow K$
set local_model $\leftarrow Model$
set global_model $\leftarrow Model$
for *dataset in* {*Yearly,Quarterly,Monthly,Weekly,Daily,Hourly*} **do**
  initiate data and parameters
  set $trainset \leftarrow read\_trainset(dataset)$
  set $testset \leftarrow read\_testset(dataset)$
  set $tsfeatures \leftarrow read\_tsfeatures(dataset)$
  set $frequency \leftarrow set\_frequency(dataset)$
  set forecast horizon $h \leftarrow length(testset)$
  cluster timeseries features
  initiate method $cluster \leftarrow clustermethod(number\_clusters)$
  find the $clusters \leftarrow cluster.cluster\_features(tsfeatures, trainset)$
  save cluster allocation in $trainset \leftarrow clusters.clusters$
  save weights $W \leftarrow clusters.weights$
  save orders $O \leftarrow clusters.clusterids$
  fit local models
  initiate $localmodels \leftarrow LocalModel(local\_model)$
  prepare input X, output $Y \leftarrow prepare\_input\_output\_local(trainset, h)$
  fit $localmodels \leftarrow localmodels.fit(X, Y)$
  fit global model
  initiate $globalmodel \leftarrow GlobalModel(global\_model, localmodels, O, W, h)$
  prepare input X, output $Y \leftarrow prepare\_input\_output\_global(trainset, h)$
  fit $globalmodel \leftarrow globalmodel.fit(X, Y)$
  save LESST model
  save $LESST \leftarrow save(globalmodel, localmodels, clusters)$
  make forecasts using LESST model
  predictions $\leftarrow LESST.predict(trainset)$
end

4.1.15 Global Model: Weighted Sum

Instead of using the methods discussed in section 4.1.4 for the Global Model, it also possible to simply take the weighted sum of the forecasts of all the Local Models. The weighted sum can be taken in different ways, we consider two of those.

Firstly, we consider simply weighting the Local Model forecasts with the weights W that we produced in the clustering step. Subsequently, we add up the weighted forecasts of all the

Local Models to reach one forecast. The weighted sum is performed as shown in equation 16

$$y_{t_i+q} = w_{i,1}\mathcal{L}(y_{t_i+q}|Y^{(1)}) + w_{i,2}\mathcal{L}(y_{t_i+q}|Y^{(2)}) + ... + w_{i,K}\mathcal{L}(y_{t_i+q}|Y^{(K)}) \tag{16}$$

Here $i = 1, 2, ..., N$ is the timeseries considered, $q = 1, 2, ..., h$ is the forecast step ahead and K is the number of clusters considered. Furthermore, $w_{i,k}$ is the weight for series i and cluster k and $\mathcal{L}(y_{t_i+q}|Y^{(k)})$ is the Local Model q step ahead forecast for cluster k of series i.

The second method we consider for taking the weighted sum is simply the even weighted sum of the forecasts of the Local Models. We can consider the same equation 16 by simply setting all the weights $w_{i,k} = 1/K$, where K is the total number of Local Models and clusters.

We do not use the weighted sum method in the Global Model for the main results of LESST. However, we consider to use it for comparison with the Global Models of LESST.

4.2 Benchmark model

The Theta model proposed by Assimakopoulos & Nikolopoulos (2000) is a univariate forecasting method that uses a theta θ coefficient to modify the curvature of a timeseries. The model is capable of producing accurate forecasts, which has been proven during the M3 forecasting competition. In this competition the method reached first place and since has been frequently used as a benchmark model. To compare our forecasting method with the theta model we compare the OWA of both methods, this is discussed in section 4.3.

We consider fitting the Theta model for each timeseries y_i separately, where $i \in \{1, 2, ..., N\}$. Considering an arbitrary series from our set y_i, to predict the $q = 1, 2, ..., h$ step ahead forecast we make use of the theta model in the **forecasting** package, which has been proposed by R. Hyndman & Billah (2001).

4.3 Performance Measure

As a performance measure we will be using the Overall Weighted Average (OWA), which was used in the M4 forecasting competition. The measure consists of a combination of the symmetric Mean Absolute Percentage Error (sMAPE) (Makridakis, 1993) and the Mean Absolute Scaled Error for seasonal timeseries (MASE) (R. J. Hyndman & Koehler, 2006). We implement these measures as described in Makridakis et al. (2020). We compute these

measures for each series separately within the dataset (Yearly, Quarterly, etc.) and then take the mean of the MASE and sMAPE as their final values. The calculation for the measures for series i can be viewed in equation 17 and 18. To calculate the total MASE and sMAPE we simply take the average for all series $\frac{1}{N}\sum_{i=1}^{N} MASE_i$ and $\frac{1}{N}\sum_{i=1}^{N} sMAPE_i$.

$$sMAPE_i = \frac{2}{h}\sum_{t=t_i+1}^{t_i+h}\frac{|y_{i,t} - \hat{y_{i,t}}|}{|y_{i,t}| + |\hat{y_{i,t}}|} * 100(\%) \tag{17}$$

$$MASE_i = \frac{1}{h}\frac{\sum_{t=t_i+1}^{t_i+h}|y_{i,t} - \hat{y_{i,t}}|}{\frac{1}{t_i-m}\sum_{t=m+1}^{t_i}|y_{i,t} - y_{i,t-m}|} \tag{18}$$

Here $y_{i,t}$ is the true value of the time series i at point t, $\hat{y_{i,t}}$ the estimated forecast, h the forecasting horizon, t_i the number of the data points available in-sample for series i, and m the time interval (frequency) between successive observations considered for each data frequency: 12 for monthly, 4 for quarterly, 24 for hourly, 52 for weekly and 1 for yearly and daily data (Makridakis et al., 2020). To calculate the OWA we need to compute the sMAPE and MASE for the Naïve2 method. Likewise, these are calculated for each series separately and then aggregated by taking the mean. Consequently we calculate the OWA as follows:

$$OWA = \frac{\frac{sMAPE(model)}{sMape(naïve2)} + \frac{MASE(model)}{MASE(naïve2)}}{2} \tag{19}$$

Viewing equation 19 we see that the OWA will be of value 1 in the case where the *model* is just as good as the *naïve*2 method. If the OWA is below 1, the *model* performs better than the *naïve*2 and if it is above 1 the model performs worse. the result section 5 we will compare the OWA of LESST with that of the Theta model to evaluate the performance difference. Furthermore, to show whether the Global Model improves the Local Model predictions we compare our main results with those of the Local Model predictions. Furthermore, we compare the performance of the combination of weighted Local Model predictions with those of the Global Model. As an additional performance measure, we will include the RMSE. The RMSE calculation for series i is given in equation 20. To calculate the total RMSE we take

the average for all series $\frac{1}{N}\sum_{i=1}^{N} RMSE_i$.

$$RMSE_i = \sqrt{\frac{1}{h}\sum_{t=t_i+1}^{t_i+h}(y_{i,t} - \hat{y_{i,t}})^2} \tag{20}$$

Here $y_{i,t}$ is the value of the time series at point t for series i, $\hat{y_{i,t}}$ the estimated forecast and h the forecasting horizon. As our main method, benchmark and performance measures have been discussed, we now move on to the result section 5.

5 Results

In this section we will first give an overview on the results of our main method LESST, which is covered in section 5.1. Moving along, we dive further into the performance of the Theta model and compare these with our results for the LESST model in section 5.2. Lastly, we discuss the Local and Global Model performances of LESST, covering some additional Global Model methods in section 5.3.

5.1 LESST model results

Using the LESST model we have obtained results for the case where the data was not deseasonalized at all and the case where series with seasonality were deseasonalized. In table 3 we show the OWA results of LESST for non-deseasonalized series having used the Yearly dataset. On the left axis of the table we see the models used in the LESST method, here the left is the method used for the Local Model and the right is the Global Model method. The number of clusters used in the model can be viewed in the columns of the table.

Table 3: Yearly OWA results LESST for non-deseasonalized data

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	1.184	1.190	1.178	1.176	1.177
Huber-Huber	0.965	0.958	0.967	0.968	0.953
Xgb-Xgb	1.172	1.159	1.150	1.146	1.144
Lgbm-Lgbm	1.153	1.152	1.139	1.133	1.131
Huber-Rf	1.240	1.147	1.109	1.102	1.096
Lgbm-Huber	1.033	1.077	1.090	1.092	1.054

[1] This table contains the OWA scores for LESST using the Yearly dataset without deseasonalization. The Local-Global model combination is presented by the most left column and all other columns represent the number of clusters used in fitting the LESST model

Viewing table 3 we see that LESST does not perform better than the Naïve2 method in most cases except for the Huber-Huber method, according to the OWA, as all other values are above 1. In table 3 the best OWA score is marked dark green, for any other model combination the best OWA score is marked light green and the worst OWA score is marked red.

It is visible from the table that the model improves with a higher number of clusters for most model combinations. The only case where this doesn't apply is the Lgbm-Huber method. For this exception, the number of clusters was 3 with which it obtained an OWA of 1.033. LESST performed its best using the Huber method as both the Local and Global model, obtaining an OWA of 0.953. The OLS however, performed the worst out of all the Local-Global model options. The OWA for this model combination seems to be much higher than that of the other models where the worst OWA obtained was 1.184.

We now move on to the results of LESST for the deseasonalized Yearly series, these can be viewed in table 4. The structure of the table is the same as in table 3. The way of marking OWA values using colors is also the same, however one row is completely marked red as the OWA values are terrible.

Table 4: Yearly OWA results LESST for deseasonalized data

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	3.192	3.131	3.122	3.177	3.542
Huber-Huber	1.105	1.104	1.106	1.113	1.114
Xgb-Xgb	1.225	1.340	1.424	1.387	1.376
Lgbm-Lgbm	1.332	1.505	1.519	1.458	1.507
Huber-Rf	1.251	1.146	1.118	1.106	1.102
Lgbm-Huber	1.7346	1.554	1.320	1.210	1.213

[1] This table contains the OWA scores for LESST using the Yearly dataset with deseasonalization. The Local-Global model combination is presented by the most left column and all other columns represent the number of clusters used in fitting the LESST model

The OWA results in table 4 seem fairly different from those in table 3 at a first glance. The OWA values are overall higher than in the previous table. The Local-Global model combination using the OLS does not appear to be the right model for this method, as the OWA values are significantly higher. The differences in OWA between model combinations and cluster numbers are larger than in the non-deseasonalized case. It therefore seems that the model using the deseasonalized series was a bit more sensitive to parameter tuning as compared to when using the non-deseasonalized series. For the deseasonalized Yearly series there is not a clear improvement among the number of clusters. Lastly, we see that the Local-Global model combination that obtained the best result is the Huber-Rf method using a 100 clusters, resulting in an OWA of 1.102

Having looked at both the difference between the number of clusters and model methods, we now summarize the best results over all the datasets in table 5. In table 5 the results of the best Local-Global model and cluster combination are used, using the LESST model trained on non-deseasonalized series. Comparing the results for all datasets, it appears that LESST performs the best in terms of OWA for the Yearly dataset, with an OWA of 0.953. The worst OWA score was obtained for the Hourly dataset, with an OWA of 4.407. This is a very high OWA score, which could be due to the data not being deseasonalized.

Table 5: Best LESST results for non-deseasonalized series

	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
OWA	0.953	1.004	1.110	1.194	1.009	4.407
SMAPE	15.118	11.167	15.436	9.655	3.084	34.612
MASE	3.900	1.363	1.224	0.703	1.174	16.600
RMSE	1034.509	738.820	805.962	440.600	215.530	774.176

[1] This table contains the best scores for all performance measures, using the LESST model for series without deseasonalization. The performance measure is given in the most left column and all other columns represent the dataset used.

Concerning the SMAPE and MASE, it can be seen that the scores are the lowest for the Hourly dataset as well. For the SMAPE the Daily set obtains the best and for the MASE the Weekly dataset has a better score. Lastly, comparing the RMSE scores we see that the model using the Yearly dataset has a much higher OWA value. Here the daily set again has the lowest value. Comparing LESST with the Naïve2 using the OWA values, we see that in this case only the LESST model using the Yearly dataset performs better. However, LESST does get close for the Quarterly and Daily dataset, as these are nearly 1.

We now move on to the best performance measure results for LESST trained with deseasonalization. The results are visible for each dataset used in table 6.

Table 6: Best LESST results for deseasonalized series

	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
OWA	1.102	0.953	1.130	1.160	1.045	0.929
SMAPE	17.020	10.657	15.118	9.383	3.194	18.144
MASE	4.618	1.286	1.289	0.683	1.217	2.085
RMSE	1072.467	698.772	768.130	481.629	219.249	482.247

[1] This table contains the best scores for all performance measures, using the LESST model for series with deseasonalization. The performance measure is given in the most left column and all other columns represent the dataset used.

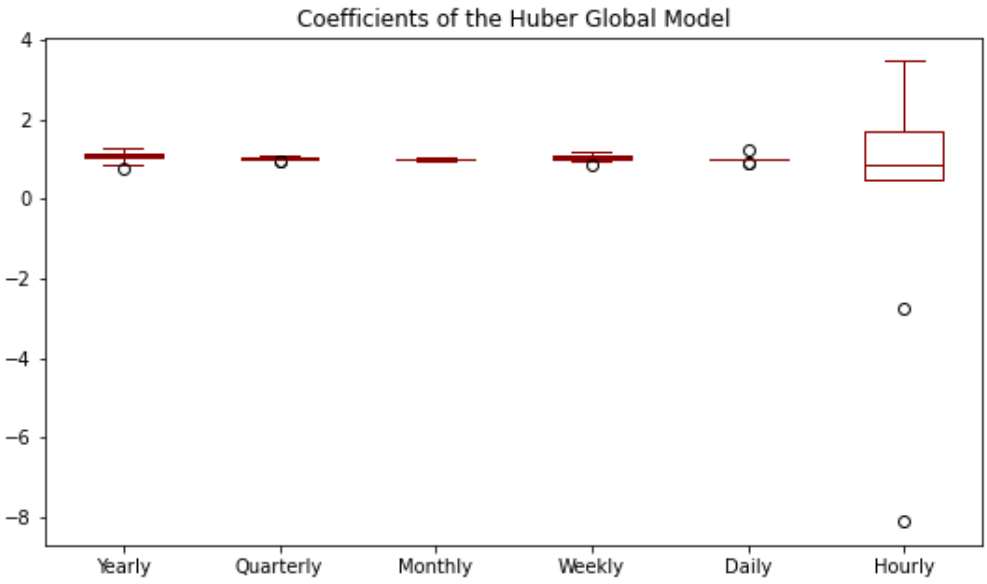
For the LESST results concerning deseasonalized series, it is visible that the method performs the worst for the Weekly dataset but not in all aspects of performance measures. Similar to the previous results, we see that LESST is more accurate for the datasets with a higher frequency. Overall the best results are very similar to the case where LESST was not deseasonalized. However, in this case we see that LESST improves a lot for the Hourly and Quarterly dataset using deseasonalization.

Furthermore, the result for the Weekly dataset seem to slightly improve using the deseasonalization. With an OWA score of 0.923 and 0.953 for the Hourly and Quarterly dataset respectively, we see that LESST outperforms the Naive2 method as the OWA value is below 1. The good hourly OWA score could be a result of the MASE being much lower than that of the Naive2 method, as the SMAPE is scoring worse compared to the other datasets with an SMAPE of 18.144.

The Yearly dataset obtains once again the highest RMSE and additionally the highest MASE with the values being 1072.467 and 4.618 respectively. Overall, comparing the LESST results for the deseasonalized case with the non-deseasonalized, the deseasonalized results seem more consistent. However, it is still quite dependent on the dataset and which parameters are chosen for the model.

To give a short overview of how the Global Model combines the weighted Local Model forecasts we show the coefficients of the Huber model as Global Model. We show the Huber coefficients for the Global Model for all datasets using 10 clusters and without deseasonalizing the data in figure 1.

Figure 1: Coefficients of the Huber Global Model



[1] This box plot figure represent the values of the 10 coefficients for the LESST Global Model using the Huber model and 10 clusters for each dataset.

From figure 1 it is visible that often the coefficients are close to 1, this is the case for five out of the six datasets. For these datasets the coefficients only slightly deviate from the value one. Solely for the Hourly dataset the coefficients of the Global Model are significantly different, as both negative and positive coefficients are present. Furthermore, the coefficients are much larger or smaller in size compared to the coefficients of the other datasets, which could indicate more instability in the Global Model. This doesn't necessarily have to lead to instability as the weights with which the Local Models predictions are weighted differ across models. However, table 5 confirms that this is the case, as the OWA is significantly larger for the Hourly dataset.

5.2 Theta and LESST performance comparison

Moving on, we now inspect the results of the Theta model, our benchmark model. The results for the Theta model can be viewed in table 7. Again we consider inspecting the OWA, SMAPE, MASE and RMSE. It can be easily seen that the Theta model often outperforms the Naive2 method as the OWA is often below 1. Furthermore, the best OWA score is obtained by the model using the Yearly dataset.

The model using the Hourly dataset however, seems to perform worse. Overall, it looks as if the Theta model performs better in terms of OWA for lower frequency datasets as compared to higher ones. Although, when including the other measures this does not seem to be the case. We see that the Theta model performs worse on the Yearly dataset considering the MASE of 3.375 and the RMSE of 1020.48. The Theta model does seem to perform worse on the hourly dataset overall, also considering the SMAPE of 18.138.

Table 7: Theta performance measures

	Yearly	**Quarterly**	**Monthly**	**Weekly**	**Daily**	**Hourly**
OWA	0.870	0.917	0.907	0.948	0.998	1.006
SMAPE	14.564	10.313	13.012	7.833	3.071	18.138
MASE	3.375	1.231	0.970	0.546	1.153	2.455
RMSE	1020.480	673.151	683.716	405.175	210.368	477.616

[1] This table contains the scores for all performance measures, using the Theta model. The performance measure is given in the most left column and all other columns represent the dataset used.

Similarly, the best SMAPE, MASE and RMSE scores occur for the same datasets as the LESST model. This could be due to the scale of the data in the dataset. Hence, it might be more meaningful to compare the results between datasets using the OWA.

Concerning this, we move on to comparing all the performance measures of the best LESST model combinations in terms of the OWA with those of the Theta model in table 8.

Table 8: Best LESST and Theta performance measures

	Yearly	**Quarterly**	**Monthly**	**Weekly**	**Daily**	**Hourly**
Local	Huber	Huber	Huber	Huber	Huber	Huber
Global	Huber	Huber	Huber	RF	Huber	RF
Clusters	100	3	3	30	30	100
Deseasonalized	No	Yes	No	Yes	No	Yes
LESST OWA	0.953	0.953	1.110	1.160	1.009	0.923
Theta OWA	0.870	0.917	0.907	0.948	0.998	1.006
LESST RMSE	1018.507	698.772	805.962	481.629	215.530	482.247
Theta RMSE	1020.480	673.151	683.716	405.175	210.368	477.616
LESST SMAPE	15.118	10.657	15.436	9.383	3.084	18.114
Theta SMAPE	14.564	10.313	13.012	7.833	3.071	18.138
LESST MASE	3.900	1.286	1.224	0.683	1.174	2.085
Theta MASE	3.375	1.231	0.970	0.546	1.153	2.455

[1] This table contains the best model combinations of LESST based on the OWA. The model specification is given along with the performance measures and the benchmark performance measures of the Theta model.

We see that in most cases the Huber method performs the best for the LESST method. However, solely using the Huber method does not appear to be sufficient for outperforming the Theta model. Accompanying the Huber method in the Local model with the Random Forest method in the Global model however, seems to do the trick. Using this combination, LESST outperforms the Theta model with an OWA of 0.923 as opposed to 1.006 for the Hourly dataset. Overall, it is not the case that the performance of LESST is miles off compared to the Theta model, but there is still a visible difference. LESST does get close with the Theta model for the Daily, Quarterly and Yearly dataset, considering all the performance measures available. Additionally, the results for the Quarterly dataset, do not seem to be far off. The worst performing instance of LESST is that of the Weekly dataset with an OWA of 1.160. Concerning the other performance measures, LESST seems to score a slightly lower RMSE of 1018.507 for the Yearly dataset. Furthermore, the SMAPE and MASE are also

better for LESST using the Hourly dataset, with values of 18.114 and 2.085 respectively.

5.3 Local and Global model performance

In this section we give an overview of the performance of the Local and Global Models of LESST. For the evaluation of the performance the same performance measures are shown. We compare the results to both the results of the Theta performance (benchmark) and LESST performance.

Firstly, we start by showing the Local Model results in table 9. These are the results of the Local Models of the best performing LESST models. The predictions are produced for each subset of timeseries separately using the Local Model belonging to the subset.

Table 9: Local Model and Theta performance

	Yearly	**Quarterly**	**Monthly**	**Weekly**	**Daily**	**Hourly**
Local OWA	0.917	0.982	1.098	0.999	0.961	0.995
Theta OWA	0.870	0.917	0.907	0.948	0.998	1.006
Local RMSE	1010.222	702.595	802.384	421.861	204.892	407.966
Theta RMSE	1020.480	673.151	683.716	405.175	210.368	477.616
Local SMAPE	14.681	10.847	15.327	8.221	2.966	16.343
Theta SMAPE	14.564	10.313	13.012	7.833	3.071	18.138
Local MASE	3.723	1.342	1.206	0.578	1.107	2.638
Theta MASE	3.375	1.231	0.970	0.546	1.153	2.455

[1] This table contains the local model results of the best model combinations of LESST based on the OWA. All performance measures and the benchmark performance measures of the Theta model are given.

The Local model obtains an OWA of 0.961 and 0.995 for the Daily and Hourly dataset respectively. Whereas, the Theta model obtains an OWA of 0.998 and 1.006. Clearly the Local Model outperforms the Theta model for the Daily and Hourly dataset, even if we consider all the performance measures. Solely the MASE of the Theta model is slightly lower compared

to that of the Local Model. The Local Model performs closely with the Theta model for the other datasets with the Monthly as exception. The OWA for the Monthly series is 1.098 for the Local Model, which is much larger than the 0.907 obtained by the Theta Model. What is more, is that the difference among all other performance measures for the Monthly series is large. Hence, the Theta heavily outperforms the Local Model for the Monthly series. As the number of clusters is only three for the Local Model of the Monthly series, it could be possible that the performance is better for a higher number of clusters. Similarly to LESST, the Local Model performs better than the Theta model in terms of RMSE. The RMSE obtained for the Local Model values at 1010.222, whereas the Theta model obtains 1020.480 in the RMSE. Seeing that the OWA is lower than 1 for most datasets, the Local Model does perform better than the Naive2 method with exception of the Monthly series.

Next, we move to the Global Model performance. For comparison with the Global Model of LESST, we consider the weighted sum of the Local Model forecasts, which we denote as Global^{sum}. Another comparison can be made for the Local Model forecasts which are weighted evenly, we denote this model by Global^{even}. Lastly, as discussed in section 4 we extend the LESST model by incorporating the whole span of data into the Global Model, which we denote as LESST^{+}. The results are made available in table 10.

Table 10: Global Model performance

	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
LESST OWA	0.953	0.953	1.110	1.160	1.009	0.923
LESST^{+} OWA	0.911	0.965	1.109	1.302	0.961	1.695
Global^{sum} OWA	0.912	0.991	1.100	4.977	0.961	152.878
Global^{even} OWA	0.903	0.993	1.099	4.487	0.951	208.279
LESST RMSE	1018.507	698.772	805.962	481.629	215.530	482.247
LESST^{+} RMSE	1040.834	703.658	805.980	536.501	206.477	429.894
Global^{sum} RMSE	1002.356	704.192	804.452	1688.709	205.647	1716.898
Global^{even} RMSE	1008.423	704.635	804.106	1432.296	203.972	1984.330

[1] This table contains the Global Model results of the best model combinations of LESST based on the OWA. LESST^{+} is an extension of LESST using more data in the Global Model. Global^{sum} and Global^{even} are the Global Models that use the sum of weighted Local Model predictions as Global model predictions, the first takes the weights derived from clusters and the latter an even weight for all models.

Comparing the Global Model results of LESST with LESST^{+}, we see that the OWA is slightly improved for the Yearly, Monthly and Daily dataset, lowering the OWA from 0.953 to 0.911, 1.110 to 1.109 and from 1.009 to 0.961. However, the performance for the other datasets decreases using the LESST^{+} model. Moreover, LESST^{+} only improves performance in terms of the RMSE for the Daily and Hourly dataset. The LESST^{+} model using the Daily and Hourly dataset was unable to use the exact same model as the best model in LESST. Therefore, a Huber-Huber model was used using 3 clusters with the same seasonality option as in LESST. For the Daily dataset there was definitely an improvement made comparing the exact same model. However, for the Hourly it remained the same. LESST does consider more parameter tuning leading to its results in comparison. Furthermore, LESST has the benefit of a much shorter running time compared to LESST^{+}, making parameter tuning more accessible.

Moving on, we see that the Global^{even} and Global^{sum} model perform closely with one another.

However, the Global Model using even weights does perform slightly better in terms of OWA. Overall, to argue which performs better is difficult as it clearly depends on the dataset and which measure is used.

When comparing the results of the Global Model of LESST with these models, we see that LESST obtains better results for halve of the datasets in terms of both the OWA and RMSE. These datasets include the Quarterly, Weekly and Hourly series for which an OWA of 0.953, 1.16 and 0.923 was achieved. Surprisingly, these are the datasets which required the series to be deseasonalized. For the datasets that did not require deseasonalization, the Global^{even} model obtained the best results. Moreover, it achieved an OWA of 0.903, 1.099 and 0.951 for the Yearly, Monthly and Daily series respectively.

Lastly, we compare the Local Model performance of LESST with the Global Model performance. We present the results for both the OWA and RMSE in table 11.

Table 11: LESST Local and Global Model performance

	Yearly	**Quarterly**	**Monthly**	**Weekly**	**Daily**	**Hourly**
LESST OWA	0.953	0.953	1.110	1.160	1.009	0.923
Local OWA	0.917	0.982	1.098	0.999	0.961	0.995
LESST RMSE	1018.507	698.772	805.962	481.629	215.530	482.247
Local RMSE	1010.222	702.595	802.384	421.861	204.892	407.966

[1] This table contains the Local and Global Model results of the best performing LESST models. Both the OWA and RMSE are considered in this table.

At a first glance, if we consider the OWA in table 11, we see that the Global Model of LESST outperforms the Local Models only for two of the datasets. When considering the RMSE, the Global Model only performs better for one dataset. Logically, the performance of the Global Model is linked with those of the of the Local Models. It is impossible for the Global Model to perform well if the Local Models perform poorly. However, from table 11 we do see that it is possible to further improve the Local Model predictions in some cases. Oddly, both the Quarterly and Hourly datasets, were the series requiring deseasonalization for better performance.

6 Discussion

Considering all the results previously discussed, we can conclude that our method LESST was unable to outperform the Theta Model. Only for the Hourly dataset was LESST outperforming the Theta model. Although, our method is at least performing competitively. The dataset with which LESST struggled the most was the Monthly dataset. Furthermore, by inspecting the Local Models, we see that the Global Model is not always capable of obtaining better results. However, in some cases the Global Model is capable of further improving the predictions of the Local Models. Moreover, it seems that for some cases the Local Models are not optimally combined using the Global Model, resulting in worse predictions. From the results it was also visible that deseasonalization of the dataset was not always necessary for it to perform well in the LESST model. Lastly, weighting Local Models with the cluster produced weights as opposed to using even weights, performed closely with one another for the weighted sum Global Model method. This indicates that the weights obtained from the clusters are relatively solid.

All in all, we were capable of producing a competitive model using the framework of the Learning with Subset Stacking model (Birbil et al., 2021) using timeseries data.

Due to many datasets, steps in the method and the variety of methods used, we were unable to dive deeply into why certain methods do not work for specific cases. Furthermore, the hyper parameter tuning for the methods used in the Local and Global Models was limited due to time constraints. Moreover, using the Global Model with all the data available in the train set is heavily taxing. Too many data points are considered in this model making it highly demanding on the computer system. Even though this method improved performance occasionally, it does not seem to be worth the additional computational cost for larger datasets. In our research we computed the timeseries features before the data was deseasonalized, doing this differently might affect the results for the case where LESST was used with the deseasonalized series.

For future research there are several possible extensions. Firstly, different weighting schemes can be considered, for example an exponential decline in the weights instead of the linear decline we use. Another approach would be to form clusters of the Local Models and

either disregard the Global Model or consider Aggregating Models for the Local Models clustered together. Furthermore, a simple extension would be to use different model methods in the Local and Global Models. As we mentioned the model methods we included lacked hyper parameter tuning, it might be something that could be included in future research. In our research we mentioned that there were some covariates given with the M4 dataset that defined the industry of the timeseries, possibly these covariates could be used to extend the LESST model. Considering a different cluster method is also an option, this could either be a different cluster algorithm or using something else instead of the timeseries features we use. Furthermore, one could consider to perform the deseasonalization before the calculation of the timeseries features, to see whether it would improve the LESST results for the model using the deseasonalized series. Since the LESST model is easily implemented with different model methods, it is also possible to combine these different LESST models into one by combining their predictions.

References

Assimakopoulos, V., & Nikolopoulos, K. (2000, 10). The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, *16*, 521-530. doi: 10.1016/S0169-2070(00)00066-2

Bentley, J. L. (1975, sep). Multidimensional binary search trees used for associative searching. *Commun. ACM*, *18*(9), 509–517. Retrieved from `https://doi.org/10.1145/361002.361007` doi: 10.1145/361002.361007

Birbil, I., Yıldırım, S., Gökalp, K., & Akyuz, H. (2021, 12). *Learning with subset stacking.*

Breiman, L. (2001, 10). Random forests. *Machine Learning*, *45*, 5-32. doi: 10.1023/A:1010950718922

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In (p. 785–794). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2939672.2939785` doi: 10.1145/2939672.2939785

Fiorucci, J. A., Pellegrini, T. R., Louzada, F., Petropoulos, F., & Koehler, A. B. (2016). Models for optimising the theta method and their relationship to state space models. *International Journal of Forecasting*, *32*(4), 1151-1161. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0169207016300243` doi: https://doi.org/10.1016/j.ijforecast.2016.02.005

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Retrieved from `http://www.jstor.org/stable/2699986`

Godahewa, R., Bandara, K., Webb, G., Smyl, S., & Bergmeir, C. (2021, 09). Ensembles of localised models for time series forecasting. *Knowledge-Based Systems*, *233*, 107518. doi: 10.1016/j.knosys.2021.107518

Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). *Why do tree-based models still outperform deep learning on tabular data?*

Hyndman, R., & Billah, B. (2001, June). Unmasking the theta method.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679-688. Retrieved from https://www.sciencedirect.com/science/article/pii/S0169207006000239 doi: https://doi.org/10.1016/j.ijforecast.2006.03.001

Kaggle. (2020). *M4 dataset.* Retrieved from https://www.kaggle.com/datasets/yogesh94/m4-forecasting-competition-dataset?resource=download&select=m4_info.csv (Accessed: 2022-06-24)

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Liang, M., Wang, X., & Wu, S. (2021, 06). A novel time-sensitive composite similarity model for multivariate time-series correlation analysis. *Entropy*, *23*, 731. doi: 10.3390/e23060731

Louppe, G. (2014). *Understanding random forests: From theory to practice* (Doctoral dissertation). doi: 10.13140/2.1.1570.5928

Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q., & Niu, X. (2018, 08). Study on a prediction of p2p network loan default based on the machine learning lightgbm and xgboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, *31*. doi: 10.1016/j.elerap.2018.08.002

Makridakis, S. (1993). Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, *9*(4), 527-529. Retrieved from https://www.sciencedirect.com/science/article/pii/0169207093900793 doi: https://doi.org/10.1016/0169-2070(93)90079-3

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018, March). Statistical and machine learning forecasting methods: Concerns and ways forward. , *13*, e0194889. doi: 10.1371/journal.pone.0194889

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54-74. (M4 Competition) doi: https://doi.org/10.1016/j.ijforecast.2019.04.014

Montero-Manso, P., Athanasopoulos, G., Hyndman, R., & Talagala, T. (2019, 09). Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, *36*. doi: 10.1016/j.ijforecast.2019.02.011

Semenoglou, A.-A., Spiliotis, E., Makridakis, S., & Assimakopoulos, V. (2020, 12). Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, *37*. doi: 10.1016/j.ijforecast.2020.11.009

Smyl, S. (2017, 06). Ensemble of specialized neural networks for time series forecasting..

Smyl, S. (2019, 07). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*. doi: 10.1016/j.ijforecast.2019.03.017

Wang, X., Hyndman, R., Li, F., & Kang, Y. (2022, 05). *Forecast combinations: an over 50-year review.*

Yin, Y., Ng, K. H., & Eam, L. (2012, 01). The relative predictive ability of forecast weight averaging and model averaging procedure. *Economic Computation and Economic Cybernetics Studies and Research*, *46*, 213-230.

Zhao, S., & Feng, Y. (2020, 01). *For2for: Learning to forecast from forecasts.*

7 Appendix

7.1 A.1: All OWA results LESST

In this section we show all LESST results concerning the OWA performance measure. For cases where there is an empty value in the table, the method was unable to train the model. This was caused by the clustering and smaller datasets causing errors in the Huber method.

7.1.1 Deseasonalized

Table 12: Yearly OWA deseasonalized

	3 clusters	**10 clusters**	**30 clusters**	**50 clusters**	**100 clusters**
Ols-Ols	3.192	3.131	3.122	3.177	3.542
Huber-Huber	1.105	1.104	1.106	1.113	1.114
Xgb-Xgb	1.225	1.340	1.424	1.387	1.376
Lgbm-Lgbm	1.332	1.505	1.519	1.458	1.507
Huber-Rf	1.251	1.146	1.118	1.106	1.102
Lgbm-Huber	1.735	1.554	1.320	1.210	1.213

Table 13: Quarterly OWA deseasonalized

	3 clusters	**10 clusters**	**30 clusters**	**50 clusters**	**100 clusters**
Ols-Ols	10.486	10.204	10.556	9.986	9.407
Huber-Huber	0.953	1.187	1.173	1.041	1.017
Xgb-Xgb	1.385	1.630	1.601	1.660	1.744
Lgbm-Lgbm	1.702	2.626	2.583	2.744	2.656
Huber-Rf	1.321	1.151	1.116	1.108	1.100
Lgbm-Huber	1.491	1.502	1.523	1.406	1.263

Table 14: Montlhy OWA deseasonalized

	3 clusters	**10 clusters**	**30 clusters**	**50 clusters**	**100 clusters**
Ols-Ols	44.369	37.736	36.214	35.067	34.342
Huber-Huber	1.130	1.838	1.809	1.241	1.324
Xgb-Xgb	1.452	1.810	1.792	1.807	1.824
Lgbm-Lgbm	1.918	3.174	3.480	3.653	3.762
Huber-Rf	1.508	1.258	1.234	1.221	1.221
Lgbm-Huber	1.254	1.393	1.480	1.418	1.367

Table 15: Weekly OWA deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	2.413	25.479	16.424	12.629	23.245
Huber-Huber	1.708	7.372	4.525	7.830	6.735
Xgb-Xgb	1.643	1.746	1.833	1.811	2.002
Lgbm-Lgbm	2.197	2.555	3.482	4.504	2.842
Huber-Rf	1.388	1.372	1.160	1.232	1.317
Lgbm-Huber	4.727	7.079	11.181	11.909	16.412

Table 16: Daily OWA deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	3.330	6.426	10.145	5.037	5.488
Huber-Huber	1.045	1.066	1.063	1.052	
Xgb-Xgb	2.428	1.680	1.603	1.617	1.593
Lgbm-Lgbm	2.918	2.195	1.963	1.843	1.878
Huber-Rf	1.312	1.339	1.267	1.218	
Lgbm-Huber	7.581	3.215	2.073	3.198	1.696

Table 17: Hourly OWA deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	14.207	35.008	17.246	45.299	27.739
Huber-Huber	1.695	3.192	3.118	4.816	9.888
Xgb-Xgb	2.283	3.167	4.556	5.249	6.019
Lgbm-Lgbm	6.053	6.240	11.668	5.883	10.749
Huber-Rf	1.289	1.084	0.961	1.008	0.929
Lgbm-Huber	4.286	9.293	762.008	187.876	276.495

7.1.2 Non-deseasonalized

Table 18: Yearly OWA non-deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	1.184	1.190	1.178	1.176	1.177
Huber-Huber	0.965	0.958	0.967	0.968	0.953
Xgb-Xgb	1.172	1.159	1.150	1.146	1.144
Lgbm-Lgbm	1.153	1.152	1.139	1.133	1.131
Huber-Rf	1.240	1.147	1.109	1.102	1.096
Lgbm-Huber	1.033	1.077	1.090	1.092	1.054

Table 19: Quarterly OWA non-deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	1.349	1.350	1.358	1.362	1.363
Huber-Huber	1.004	1.005	1.025	1.024	1.032
Xgb-Xgb	1.182	1.234	1.221	1.222	1.223
Lgbm-Lgbm	1.158	1.224	1.217	1.204	1.202
Huber-Rf	1.349	1.206	1.168	1.162	1.156
Lgbm-Huber	1.116	1.147	1.202	1.175	1.158

Table 20: Monthly OWA non-deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	1.811	1.824	1.828	1.811	1.805
Huber-Huber	1.110	1.112	1.125	1.133	1.127
Xgb-Xgb	1.382	1.410	1.401	1.396	1.393
Lgbm-Lgbm	1.358	1.427	1.417	1.397	1.381
Huber-Rf	1.678	1.414	1.377	1.370	1.366
Lgbm-Huber	1.206	1.210	1.290	1.285	1.243

Table 21: Weekly OWA non-deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	1.680	1.596	1.601	1.635	1.679
Huber-Huber	1.271	1.330	1.194	1.204	1.320
Xgb-Xgb	1.503	1.530	1.489	1.434	1.514
Lgbm-Lgbm	1.542	1.512	1.489	1.417	1.449
Huber-Rf	1.418	1.396	1.246	1.260	1.237
Lgbm-Huber	2.114	1.768	1.925	2.036	1.959

Table 22: Daily non-deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	1.520	1.174	1.126	1.145	1.148
Huber-Huber	1.020	1.021	1.009	1.021	
Xgb-Xgb	1.400	1.591	1.521	1.504	1.489
Lgbm-Lgbm	1.371	1.666	1.655	1.630	1.589
Huber-Rf	1.317	1.338	1.269	1.227	
Lgbm-Huber	1.578	1.365	1.428	1.500	1.553

Table 23: Hourly OWA non-deseasonalized

	3 clusters	10 clusters	30 clusters	50 clusters	100 clusters
Ols-Ols	77.199	273.541	177.358	85.981	95.671
Huber-Huber	6.910	12.461		14.072	
Xgb-Xgb	19.361	19.922	18.741	17.925	17.065
Lgbm-Lgbm	29.025	32.770	24.985	39.220	41.184
Huber-Rf	11.930	4.958		4.407	
Lgbm-Huber	10.704	35.275	35.070	271.264	192.286

7.2 A.2: Time series features

	Feature	Description
1	length	length of time series
2	trend	strength of trend
3	seasonality	strength of seasonality
4	linearity	linearity
5	curvature	curvature
6	spikiness	spikiness
7	e_acf1	first ACF value of remainder series
8	e_acf10	sum of squares of first 10 ACF values of remainder series
9	stability	stability
10	lumpiness	lumpiness
11	entropy	spectral entropy
12	hurst	Hurst exponent
13	nonlinearity	nonlinearity
14	alpha	ETS(A,A,N) $\hat{\alpha}$
15	beta	ETS(A,A,N) $\hat{\beta}$
16	hwalpha	ETS(A,A,A) $\hat{\alpha}$
17	hwbeta	ETS(A,A,A) $\hat{\beta}$
18	hwgamma	ETS(A,A,A) $\hat{\gamma}$
19	ur_pp	test statistic based on Phillips-Perron test
20	ur_kpss	test statistic based on KPSS test
21	x_acf1	first ACF value of the original series
22	diff1_acf1	first ACF value of the differenced series
23	diff2_acf1	first ACF value of the twice-differenced series
24	x_acf10	sum of squares of first 10 ACF values of original series
25	diff1_acf10	sum of squares of first 10 ACF values of the differenced series
26	diff2_acf10	sum of squares of first 10 ACF values of the twice differenced series
27	seas_acf1	autocorrelation coefficent at first seasonal lag
28	diff2x_pacf5	sum of squares of first 5 PACF values of twice-differenced series
29	seas_pacf	partial autocorrelation coefficient at first seasonal lag
30	crossing_point	number of times the time series crosses the median
31	flat_spots	number of flat spots
32	nperiods	number of seasonal periods inthe series
33	seasonal_period	length of seasonal period
34	peak	strength of peak
35	trough	strength of trough
36	arch_acf	sum of squares of the first 12 autocorrelations of z^2
37	garch_acf	sum of squares of the first 12 autocorrelations of r^2
38	achr_r2	R^2 value of an AR model applied to z^2
39	garch_r2	R^2 value of an AR model applied to r^2
40	month	Month of the year
41	time interval	time interval (daily/hourly/15 min interval)

Table 24: All time series features

7.3 A.3: Repositories

7.3.1 Git Repositories

Forecast R:`https://github.com/robjhyndman/forecast`

LESS: `https://github.com/sibirbil/LESS`

TSfeatues: `https://github.com/Nixtla/tsfeatures`

ESRNN: `https://github.com/AlexDowney/ESRNN_fork`

FFORMA and Rforecast-models: `https://github.com/christophmark/fforma`

LESST: `https://github.com/Stephen97T/LESST`

7.4 A.4: Code description

In this section we describe the details on the code, what files contain what and how to run the results: Follow these steps if running the code for the first time:

- Install R
- Install all required python libraries in the requirements.txt into the python environment
- Change the path in $os.environ["R\_HOME"] = "E : /documents/work/mini/envs/work/lib/R"$ in the preprocessing.py and tsforecast.py files to the path of your python environment
- Uncomment the R library installations in the preprocessing file
- Run the prepare_allm4data function in the preprocessing.py file
- Run the prepare_m4tsfeatures function in the preprocessing.py file for calculating all the timeseries features of all dataset and saving them
- Comment the R library installations in the preprocessing file again

For obtaining the results run theses files:

- ResultsMain.py: main LESST results and benchmark performance
- ResultsLocal.py: performance Local Models
- ResultsGlobal.py: performance Global Model using weighted sum

- ResultsRolling.py: performance Global Model with all training data
- ResultsWeights.py: figure of Global Model coefficients
- formatresults.py: makes excel files out of the dictionaries from the main LESST results

For the dataset information run:

- DataInfo.py: calculates some information on the datasets

Here are details on what the other files contain:

- LESST.py: contains the LESST model
- Models.py: contains the Local and Global Model
- benchmark.py: contains functions for the benchmark method and the performance measures
- clustering.py: contains the timeseries feature clustering method
- seasonality: contains functions for deseasonalizing the data
- tsforecasts.py: contains the ThetaF model and other R forecasting models
- data_prep.py: contains functions for preparing input and target data
- preprocessing: contains functions for calculating timeseries features and reading m4 data