Computationally Efficient Forecasting Methods for Large-Scale Real-Time Applications

Thiyanga Shamini Talagala

M.Sc., University of Moratuwa, Sri Lanka

B.Sc. (Hons), University of Sri Jayewardenepura, Sri Lanka

A thesis submitted for the degree of Doctor of Philosophy at Monash University in 2019

Department of Econometrics and Business Statistics

Contents

Copyright notice

Abstract

Forecasting is a key activity for any business to operate efficiently. The dramatic increase in the availability of large collections of time series raises the need for developing reliable efficient and automatic algorithms for forecasting. This thesis presents three such algorithms for large-scale applications based on the meta-learning approach. The key idea behind the methodology is the use of a vector of features on each time series, measuring global characteristics of the series. The process starts with an offline phase: train a meta-learner to relate the features of time series to the performance of different forecast models using a large historical collection of time series. The online phase of generating forecasts involves only the calculation of a simple vector of features for any newly given time series and the application of a pre-trained meta-learner to produce the required forecasts.

The first algorithm is FFORMS: Feature-based FORecast Model Selection. FFORMS builds a mapping that relates the features of time series to the 'best' forecast model using a random forest. The process of generating forecasts involves the application of the best model for each series. This is a very fast and reliable approach. The FFORMS algorithm is evaluated using the time series from the M1, M3 and M4 competitions and is shown to yield accurate forecasts comparable to several benchmarks and other commonly used automatic approaches for time series forecasting. Furthermore, model-agnostic machine learning interpretability tools are used to explore what is happening under the hood of the FFORMS algorithm. In particular, the thesis explores *which* features are the most important for the choice of model selection; *where* they are most important, that is, for the overall classification process or, within specific class of models or a set of multiple classes of models; *how* these features link with the predicted outcome; and *when* and *how*

strongly features interact with other features. The results provide valuable insights into how different features and their interactions affect model selection.

The second algorithm, FFORMA (Feature-based FORecast Model Averaging) obtains weights for forecast combinations. The extreme gradient boost algorithm with a custom objective function is used to train a meta-model. The probabilities of each model being best are used as weights for computing a combination forecast. The FFORMA approach achieved second place in the recent M4 forecasting competition (Makridakis, Spiliotis, and Assimakopoulos, 2018).

The third algorithm is FFORMPP (Feature-based FORecast Model Performance Prediction). The efficient Bayesian multivariate surface regression approach is used to model forecast error as a function of features calculated from the time series. FFORMPP allows rankings of models according to their relative forecast performance without calculating forecasts from all available models in the pool. Further, a feature-based time series simulation approach is used to obtain a diverse collection of time series to train the meta-learner. In addition, the thesis attempts to visualise the relationships discovered by the meta-learning algorithm. The visualisation approach involves mapping each time series as a point in a two-dimensional instance space, given by the features. This helps to gain insights into why certain models perform better on certain types of time series.

Publications during enrolment

This thesis by publication is built around four articles which are at different stages of publication.

1. Chapter 2 has been submitted to the *International Journal of Forecasting* for possible publication.

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

2. Chapter 3 is ready for submission to *Computational Statistics & Data Analysis*.

   Talagala, T. S., Hyndman, R. J., & Athanasopoulos, G. (2019). Peeking inside FFORMS: Feature-based forecast model selection. Computational Statistics & Data Analysis [ready for submission].

3. Chapter 4 has been accepted for publication in the *International Journal of Forecasting* and is currently in press.

   Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., Talagala, T. S. (2019) FFORMA: Feature-based forecast model averaging. International Journal of Forecasting [to appear]

4. Chapter 5 has been submitted to the *International Journal of Forecasting* for possible publication.

   Talagala, T. S., Feng, L., & Kang, Y. (2019). FFORMPP: Feature-based forecast model performance prediction. International Journal of Forecasting [under review].

The contributions in Chapters 2 - 5 of this thesis were presented at the following events:

- 37th International Symposium on Forecasting 2017, Cairns, Australia.
- Young Statisticians Conference 2017, Tweed Heads, NSW, Australia.

  A Statistical Society of Australia travel award grant was received to attend the conference.
- Young Stats Showcase hosted by the Statistical Society of Australia, Victorian Branch, Australia, in September 2017.
- 38th International Symposium on Forecasting 2018, Boulder, Colorado, USA, in June 2018

  An International Institute of Forecasters travel award grant was received to attend the conference.
- 2018 Joint Statistical Meetings (JSM2018), Vancouver, British Columbia, Canada, in August 2018
- useR! 2018, Brisbane, Australia, in July 2018
- Invited talk, Central University of Finance and Economics, Beijing, China, in March 2019
- 39th International Symposium on Forecasting, Thessaloniki, Greece, in June 2019

  An International Institute of Forecasters travel award grant was received to attend the conference.
- useR! 2019, Toulouse, France, in July 2019

Declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes one original paper published in a peer reviewed journal and three submitted publications. The core theme of the thesis is feature-based large-scale time series forecasting. The ideas, development and writing up of all the papers (with the exception of Chapter 4) in the thesis were the principal responsibility of myself, the student, working within the Department of Econometrics and Business Statistics, Monash University, under the supervision of Professor Rob J Hyndman and Professor George Athanasopoulos.

The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into team-based research.

In the case of Chapter 2-5 my contribution to the work involved the following:

Thesis Chapter	Publication Title	Status (published, in press, accepted or returned for revision)	Nature and % of student contribution	Co-author name(s) Nature and % of Co-author's contribution	Co-author(s), Monash student Y/N
2	Meta-learning how to forecast time series	returned for revision	Formulating the approach, construction of research design, implementation, data analysis and interpretation, software development, writing the first draft (80%)	1) Rob J Hyndman, input into manuscript (10%) 2) George Athanasopoulos, input into manuscript (10%)	N
3	Peeking inside FFORMS: Feature-based forecast model selection	ready for submission	Formulating the approach, construction of research design, implementation, data analysis and interpretation, software development, writing the first draft (90%)	1) Rob J Hyndman, input into manuscript (5%) 2) George Athanasopoulos, input into manuscript (5%)	N
4	FFORMA: Feature-based forecast model averaging	published	Formulating the approach, construction of research design, initiate writing the first draft (6%)	1) Pablo Montero-Manso, input into manuscript (80%) 2) George Athanasopoulos, input into manuscript (7%) 3) Rob J Hyndman, input into manuscript (7%)	N
5	FFORMPP: Feature-based forecast model performance prediction	submitted	Formulating the approach, construction of research design, implementation, data analysis and interpretation, software development, writing the first draft (80%)	1) Feng Li, input into manuscript (10%) 2) Yanfei Kang, input into manuscript (10%)	N

I have not renumbered sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student name: Thiyanga Shamini Talagala

Student signature: **Date:** 22 August 2019

Acknowledgements

UseR 2019! in Toulouse, France, is where I gave my last talk as a PhD student. The conference gala dinner was held at the Cite' de l'espace, Toulouse. At Cite' de l'espace, I had an opportunity to experience what it feels like to be inside a true space shuttle and learn how they operate. A space shuttle needs a massive amount of energy to fight against gravitational pull, but once it has launched, the space shuttle just glides along easily. What matters for the space shuttle is the first push off the ground. Two powerful side rockets help a shuttle to leave the gravitational pull.

I am extremely thankful to my supervisors, Professor Rob J Hyndman and Professor George Athanasopoulos, for being two powerful side rockets. I thank them for giving me that first push in the right direction and for making sure I saw this project to its completion. I would like to thank them for their endless energy and guidance.

I would like to thank the PhD directors, Professor Gael Martin and Professor Farshid Vahid, for their valuable support and encouragement. I am thankful to Professor Di Cook, Dr Souhaib Ben Taieb and Dr Klaus Ackerman, the members of my PhD panel, for their constructive suggestions and helpful feedback. I am also very grateful to Professor Scott Sisson for his encouraging advice.

Chapter 5 of this thesis is a research collaboration with Associate Professor Feng Li of Central University of Finance and Economics in China and Associate Professor Yanfei Kang of Beihang University, China. I am grateful to both of them for inviting me to visit China to work on the project presented in Chapter 5. Further, I would like to thank the members of their forecasting team for their wonderful hospitality. I am grateful to Central University of Finance and Economics, Beijing, China, for providing financial support

during the visit. Further, the work presented in Chapter 4 is a research collaboration with Dr Pablo Montero-Menso. I am thankful to him for several hours of discussion on the subject.

I am very grateful to Monash University for awarding me the Monash Graduate Scholarship, and to the Faculty of Business and Economics for offering me the Faculty Postgraduate Research Scholarship. Further, I gratefully acknowledge the research funding provided by the Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers. I was also awarded two travel awards from the International Institute of Forecasters to attend the 38th International Symposium on Forecasting in Boulder, USA, and the 39th International Symposium on Forecasting in Thessaloniki, Greece. This support is gratefully acknowledged.

The computational resources and services used in this thesis were supported in part by the Monash eResearch Centre and eSolutions Research Support Services through the use of the MonARCH (Monash Advanced Research Computing Hybrid) HPC cluster. Special thanks to Philip Chen and Simon Michnowicz from the Monash E-research Centre for their time spent answering my questions and help provided in using the parallel supercomputing facility. To the members of the NUMBAT research group, the MONASH data fluency group and R-Ladies Melbourne, many thanks for making the environment a stimulating, enjoyable and supporting place for research.

I would also like to thank Elite Editing for copyediting the thesis. The editorial intervention by Elite Editing was restricted to Standards D and E of the Australian Standards for Editing Practice. I wish to extend a special thanks to the Monash Graduate Research team for their support along the way. I would also like to express my appreciation to the administrative staff of the Department of Econometrics and Business Statistics, especially Clare Livesey for her friendly support and assistance throughout!

This journey would not have been possible without the support of my family. I am extremely lucky to have done my PhD together with my sister, Priyanga Dilini Talagala. Getting full scholarships from Australia's prestigious Monash University to do our PhDs is another big accomplishment we achieved together. I thank my father and mother for their

endless encouragement and invaluable inspiration. This thesis is dedicated to my family, father, mother and sister, who have all shown me great love and continuous support in many ways all these years.

Chapter 1

Introduction

Essentially, all models are wrong, but some are useful.

— George E. P. Box, *1987*

The words of famous British statistician George Box encapsulate the essence of model selection. This statement is particularly valid for forecast model selection. For many decades, researchers have been developing increasingly sophisticated models for forecasting. From a theoretical and computational point of view, these models are elegant and efficient algorithms, which makes them appealing candidates for many applications. However, given the large number of models, it is challenging to identify which one is likely to generate the most accurate forecasts for a given time series. Even though various models for time series forecasting have been developed, little research has contributed to advice on when each model is beneficial. Moreover, as stated by Georgoff and Murdick (1986), the main issue facing forecasters is not the need for new or more sophisticated models, but the need to derive better approaches for forecast model selection. Since a trial-and-error approach is an extremely time consuming search process, researchers need to identify more efficient search strategies. The aim of this thesis is to develop new frameworks that will help in forecasting large numbers of time series completely automatically.

1.1 Background of the study

The advent of modern technologies has enabled businesses and companies to collect ever more data. Of these, time series data are everywhere and play a key role in business decision-making processes. For example, large-scale businesses, such as Facebook (Taylor and Letham, 2018) and Walmart (Seaman, 2018), need to forecast many millions of time series for capacity planning to optimise resource allocation, set targets to evaluate performance relative to baseline, and so on. Hence, accurate forecasts for these series are immensely valuable. The brute force solution is simply to try out all the candidate models on the problem at hand and select the model that performs best over the test period. This model is then used to forecast future values of the time series. However, this approach is not feasible in practice, especially when there are tens or hundreds of millions of series to forecast. The alternative solution is 'aggregate selection', in which a single method is used to compute forecasts for all time series (Fildes, 1989). According to the no-free-lunch concept, there is no single model that performs best on all kinds of time series (Wolpert and Macready, 1997). Since neither solution is ideal, there is a need for a more general and flexible framework to assist in forecasting large numbers of time series.

In an attempt to answer the question 'Which forecast model performs best on a given series?', Spyros Makridakis organised the Makridakis Competitions, also known as M-Competitions. The M-Competitions have had a profound influence on the field of time series forecasting (Hyndman, 2019). The findings of these competitions also point to many important research directions and conclusions (Makridakis and Hibon, 2001). Of these, two are particularly important for the problem addressed by this thesis. First, the characteristics of time series are useful for identifying suitable models for forecasting (Armstrong, 2001; Tashman, 2001). Second, forecasters should be able to explain the conditions under which each method works best (Hyndman, 2001; Makridakis and Hibon, 2001). To this end, this thesis investigates the utility of the characteristics of time series referred to as *features* in addressing the problem of large-scale time series forecasting. Feature-based analysis of time series has several advantages over raw point analysis: it facilitates data visualisation and understanding; it acts as a dimension-reduction tool,

thereby reducing storage requirements and computational time; it handles time series with unequal lengths (Guyon and Elisseeff, 2003); among others.

A few researchers have explored the benefit of using time series features to select suitable forecasting methods. Collopy and Armstrong (1992) introduced 99 rules based on 18 features of time series to make forecasts for economic and demographic time series. Meade (2000) uses 25 features of time series to predict the relative performance of nine different models. Petropoulos et al. (2014) examine the effect of seven time series features (seasonality, trend, cycle, randomness, length, inter-demand interval and coefficient of variation) on predicting the performance of 14 popular models through regression analysis. The authors argue for a 'horses for courses' approach to discovering the extent to which the features of time series are useful in predicting the model performance. In an attempt to answer the question raised by Hyndman (2001) in the M3 competition discussion 'why some methods work well and others do not' Kang, Hyndman, and Smith-Miles (2017) try to visualise the algorithm performance in the instance space defined by six features. A more detailed description of the literature is given in Chapter 2.

Although many researchers have highlighted the usefulness of features in forecast model selection, only a few studies have reported success against benchmarks and other commonly used approaches in forecasting. Revisiting this literature reveals significant gaps. Possible reasons for these gaps are i) an inadequate number of features were used; ii) the selected features were not useful for the purpose; iii) possible limitations in the modelling framework used (for example, linear regression models that focus on the global relationship in the data do not incorporate local relationships into the model); and iv) a lack of diversity in the training set of time series used to build the model. In this thesis, the aim is to design feature-based time series forecasting algorithms for large-scale applications and fill in the noticeable gaps in the literature.

1.2 Meta-learning

The feature-based forecast algorithms proposed in Chapters 2, 4 and 5 are based on a meta-learning approach. A framework for algorithm selection was first proposed in the seminal paper of Rice (1976). Rice's algorithm selection framework builds a mapping that

relates the features of the problem instances with the performance of different algorithms. The machine-learning community use the same idea, calling it 'meta-learning'. Although Rice's idea has been around since 1976, not many applications using his idea have been made until recently. However, with the rapid development of information technologies and the emergence of an era of big data, Rice's algorithm selection problem has attracted the attention of many disciplines and provides a promising solution to algorithm selection problems in various domains. Smith-Miles (2008) provides a comprehensive review of applications of this approach. However, despite its widespread popularity, several researchers have pointed out that it is not entirely clear from Rice's framework how the mapping is done from features to performance of different algorithms. The work was later extended and improved in a few different ways (Smith-Miles and Lopes, 2012; Kotthoff, 2014). The methodological framework used in this study is shown in Figure 1.1. The offline and online phases are shown in blue and red respectively. A meta-learner is trained during the offline phase and is then used to select an appropriate model for a new time series in the online phase. The success of a meta-learning framework depends on four main factors: i) problem space, ii) feature space, iii) algorithm space and iv) performance space. These are now described in detail.

1.2.1 Problem space

The dataset used for the study is called the problem space. In this thesis, the time series of M1, M3 (Makridakis and Hibon, 2000) and M4 competitions (Makridakis, Spiliotis, and Assimakopoulos, 2018) form the problem space. The M-competition data were a sample of time series collected from several domains such as demographics, finance, business and economics. Table 1.1 shows the classification of the M-competition series according to frequency and domain. A single time series is called *an instance* in the problem space. Prediction accuracy of a meta-learner strongly depends on the instances used to train the meta-learner. In addition to the time series from the M-Competitions, simulated time series are used to increase the diversity of the problem space. The different approaches used to simulate time series are explained in Chapters 2, 3 and 5. Further, it is important to note that instances are required that are similar to the time series that are forecasted in the online phase of the algorithm.

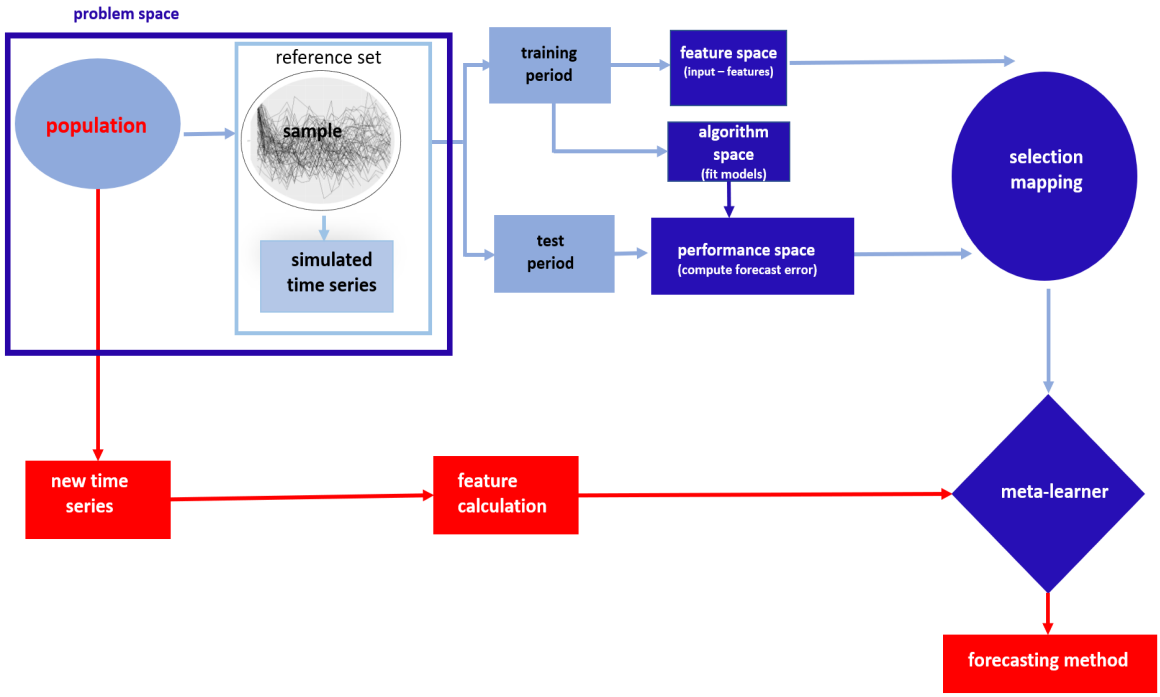

Figure 1.1: *The methodological framework. The offline phase is shown in blue and the online phase in red.*

Table 1.1: *The composition of M-competition data: the number of time series by domain and frequency category*

Period		Domain						
	Competition	Demographic	Finance	Industry	Macro	Micro	Other	Total
Yearly	M1	30	-	35	59	57	-	181
	M3	245	58	102	83	146	11	645
	M4	1088	6519	3716	3903	6538	1236	23000
Quarterly	M1	39	-	18	104	42	-	203
	M3	57	76	83	336	204	-	756
	M4	1858	5305	4637	5315	6020	865	24000
Monthly	M1	75	-	183	156	203	-	617
	M3	111	145	334	312	474	52	1428
	M4	5728	10987	10017	10016	10975	277	48000
Weekly	M4	24	164	6	41	112	12	359
Daily	M4	10	1559	422	127	1476	638	4227
Hourly	M4	0	0	0	0	0	414	414

1.2.2 Feature space

The feature space is characterised by a set of measurable characteristics of the instances in the problem space. The features can also be considered summary statistics of time series. John W. Tukey was the first to come up with the idea of 'features', which he called cognostics: computer aided diagnostics (Tukey and Tukey, 1985). A few decades later, his idea was rebranded and used under different names. A broad range of features exists that can be used to summarise time series (Fulcher and Jones, 2014). The choice of features

should take account of the final goal of the research question. Further, features of time series play a central role in both online and offline phases of the meta-learning framework. These constraints introduce the need for careful consideration of the feature selection process in the frameworks. The following properties of features are considered:

i) Informative. Most of the features introduced in the literature aim to identify different patterns in time series. In contrast in this thesis, the interest is in using features to identify the best forecasting model(s) for a given series. Hence, the features considered should provide sufficient distinction between different instances in terms of which is the most suitable model(s) for forecasting.

ii) Interpretability. The features used should provide meaningful interpretations of the instance characteristics so that practitioners can gain maximum insight into the problem space. Further, this helps explain reasons for the predictions of the meta-learner and gain trust in the proposed frameworks.

iii) Time and computational cost. Features play a central role in both offline and online phases of the algorithm. To make predictions quickly during the online phase, the features considered should be easy and quick to compute without heavy computational cost.

iv) Applicability. Features should be computable for a broad range of problem instances, rather than being restricted by different conditions, such as length, non-stationarity and other properties of time series.

Further, apart from a few exceptions (such as length of series, number of times the time series crosses the median and the number of flat spots in the series), most of the features considered here are independent of scale and are ergodic: asymptotically independent of the length of the time series. A more detailed description of the features is given in Chapters 2, 3 and 4.

1.2.3 Algorithm space

A suitable set of candidate models to include in the portfolio forms the algorithm space. In this thesis twelve methods implemented in the `forecast` package in R (Hyndman et al.,

2018) are considered for the algorithm space. This is the first time such a large collection of algorithms has been considered in a time series meta-learning framework. The algorithm space includes (The R functions are given in parentheses):

i) Random walk with drift (`rwf` with drift=TRUE)
ii) Random walk model (`rwf` with drift=FALSE)
iii) Seasonal naive (`snaive`)
iv) White noise process
v) Theta method (`thetaf`) – this method became the winner of the M3 competition (Makridakis and Hibon, 2000)
vi) TBATS (`tbats`) models introduced by De Livera, Hyndman, and Snyder (2011)
vii) Neural network forecasts (`nnetar`)
viii) automated ARIMA algorithm (`auto.arima`)
ix) exponential smooting (ETS) algorithm (`ets`)
x) STL-AR (`stlm` with model function `ar`) – seasonal and trend decomposition using loess with autoregressive (AR) modelling of the seasonally adjusted series
xi) MSTL-ETS (`mstl` with model function `ets`) – first, a multiple seasonal decomposition is applied to the time series and then the seasonal naive is used to forecast the seasonal components; then, the automated ETS algorithm is used to forecast the seasonally adjusted series.
xii) MSTL-ARIMA (`mstl` with model function `auto.arima`) – first, a multiple seasonal decomposition is applied to the time series and then the seasonal naive is used to forecast the seasonal components; then, the automated ARIMA algorithm is used to forecast the seasonally adjusted series.

The methods considered in viii)-xii), involve fully or partially use of the automated ARIMA algorithm of Hyndman and Khandakar (2008) or the automated ETS algorithm of Hyndman et al. (2002) to select the appropriate either ARIMA class model or ETS models. Next, the main steps of these algorithms are summarised.

Automated exponential smoothing (`ets`)

Exponential smoothing methods were introduced in the late 1950s (Brown, 1959; Brown, 1962). This framework was later extended by Gardner (1985) and Hyndman et al. (2002). Because of their computational simplicity and interpretability, they became widely used in practice. A classification of 15 exponential smoothing methods is presented in Table 1.2. Each model can have an additive or multiplicative error, giving 30 different models. Out of these 30 models, only 19 models are numerically stable. Further, multiplicative trend models give poor forecasts, which leaves 15 models.

Table 1.2: *Classification of exponential smoothing methods*

Trend component	Seasonal Component		
	N (None)	A (Additive)	M (Multiplicative)
N (None)	N, N	N, A	N, M
A (Additive)	A, N	A, A	A, M
A_d (Additive damped)	A_d, N	A_d, A	A_d, M
M (Multiplicative)	M, N	M, A	M, M
M_d (Multiplicative damped)	M_d, N	M_d, A	M_d, M

The steps involved are summarised below (Hyndman and Khandakar, 2008):

1. For each series, apply each of the 15 models that are appropriate for the data.
2. For each model, optimise the parameters (smoothing parameters and initial values for the states) using maximum likelihood estimation.
3. Select the best model using the corrected Akaike's Information Criterion (AICc) and produce forecasts using the best selected model.

Automated ARIMA modelling (`auto.arima`)

ARIMA is one of the most popular models for time series forecasting. One of the common difficulties in ARIMA modelling is the order selection process. Hyndman and Khandakar (2008) developed a framework to automate this process. The steps involved are summarised below: For non-seasonal data, ARIMA(p,d,q) models are considered and for seasonal data ARIMA$(p,d,q)(P,D,Q)_m$ are considered.

1. Select the number of differences d and D via unit root tests.

2. Try four possible models to start with:

i) ARIMA$(2,d,2)$ if $m = 1$ and ARIMA$(2,d,2)(1,D,1)_m$ if $m > 1$.
ii) ARIMA$(0,d,0)$ if $m = 1$ and ARIMA$(0,d,0)(0,D,0)_m$ if $m > 1$.
iii) ARIMA$(1,d,0)$ if $m = 1$ and ARIMA$(1,d,0)(1,D,0)_m$ if $m > 1$.
iv) ARIMA$(0,d,1)$ if $m = 1$ and ARIMA$(0,d,1)(0,D,1)_m$ if $m > 1$.

3. Select the model with the smallest AICc from step 2. This becomes the current model.

4. Consider up to 13 variations on the current model:

i) Vary one of p, q, P and Q from the current model by ± 1.
ii) p, q both vary from the current model by ± 1.
iii) P, Q both vary from the current model by ± 1.
iv) Include or exclude the constant term from the current model. Repeat step 4 until no lower AICc can be found. For more details, please refer to Hyndman and Khandakar (2008).

1.2.4 Algorithm performance space

Algorithm performance space is characterised by a set of metrics used to evaluate the performance of different algorithms on the instances in the problem space. Mean absolute scaled error (MASE) introduced by Hyndman and Koehler (2006) is mainly used throughout the thesis for evaluate point forecasts. Let y_t and $\hat{y}_t$ denote the observed and forecast values at time t, respectively. Then, MASE is defined by

$$\text{MASE} = \frac{\frac{1}{H}\sum_{h=1}^{H}|y_{T+h} - \hat{y}_{T+h|T}|}{\frac{1}{n-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|}.$$

where H is the length of the forecast horizon, T is the length of the training period of the time series and m is the frequency of the time series. The MASE is independent of the scale of the data. In addition to MASE, symmetric mean absolute percentage error (sMAPE) is used in Chapter 3 and Chapter 4 because the M4 competition forecasts are evaluated based on both MASE and sMAPE (*M4 Competitor's Guide* 2018). The sMAPE is simply

$$\text{sMAPE} = \frac{1}{H}\sum_{h=1}^{H}\frac{200|y_h - \hat{y}_h|}{|y_h| + |\hat{y}_h|}.$$

Prediction intervals are also computed. The performance of 95% prediction intervals are evaluated based on the mean scaled interval score (MSIS) (Gneiting and Raftery, 2007; *M4 Competitor's Guide* 2018). The MSIS is defined as

$$\text{MSIS} = \frac{\frac{1}{H}\sum_{h=1}^{H}(U_h - L_h) + 40(L_h - y_h)1\{y_h < L_h\} + 40(y_h - U_h)1\{y_h > U_h\}}{\frac{1}{n-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|},$$

where L_h and U_h denote the lower and upper bounds of the prediction intervals at time h, and 1 is the indicator function (being 1 if y_h is within the postulated intervals and 0 otherwise).

1.3 Linking feature space and algorithm performance space

Linking feature space and algorithm performance space is a central part of the meta-learning process. Three modelling approaches are used to model the relationships: i) the random forest algorithm, ii) the extreme gradient boosting algorithm and iii) the efficient Bayesian multivariate surface regression approach.

1.3.1 Random forest algorithm

The methodological framework presented in Chapter 2 is based on the random forest algorithm. A random forest (Breiman, 2001) is an ensemble learning method that combines a large number of decision trees using a two-step randomization process. This algorithm combines Brieman's idea of *bagging* and *random selection of features at each node* in each tree. The motivation behind the use of random forest classifiers are: i) it can model complex interactions between features; ii) the modelling framework captures linear and non-linear effects of features through the averaging of large number of decision trees; iii) their relative transparency; iv) its robustness against over-fitting the training data; v) it is easy to handle the problem of imbalanced classes; vi) it is a fast approach compared to boosting algorithms and vii) it is fairly easy to implement with available software.

1.3.2 Extreme gradient boosting algorithm

The methodological framework presented in Chapter 4 is based on the extreme gradient boosting algorithm. The extreme gradient boosting algorithm, also known as XGBoost, is a tree ensemble model for classification and regression introduced by Chen and Guestrin (2016). The algorithm involves fitting a sequence of weak learners (in the case of this thesis, decision trees) on reweighted data. The process starts by fitting a shallow decision tree (i.e. a weak learner) to the whole space of the training data and obtaining predictions. Then, the data are weighted according to the predictions obtained in the previous step (higher weights are assigned to misclassified training instances). The second tree is constructed based on these weighted data. This process is repeated until the stopping criterion is met. The predictions based on all trees are combined through a weighted majority vote (in the case of classification) or weighted sum (in the case of regression) to obtain the final prediction of the model. The concept is similar to the gradient boosting algorithm (Friedman, 2001) but more efficient. The XGBoost algorithm benefits from a regularised model formalisation to control overfitting. This can be formalised as follows. Let N be the total number of instances (training examples), f_i is a vector of m features for the ith data point and z_i is the ith observed value of the outcome. Then the training dataset is defined as

$$D = \{(f_i, z_i) : i = 1, \ldots, N\}, |D| = N, f_i \in \mathbb{R}^m, z_i \in \mathbb{R}.$$

The objective function of the XGBoost algorithm is

$$\mathbb{L} = \sum_{i=1}^{N} l(z_i, \hat{z}_i) + \sum_{k=1}^{K} \Omega(g_k). \tag{1.3.1}$$

This contains two parts, the loss function and the regularization term. The first term l is a differential convex loss function that measures the difference between the predicted value $\hat{z}_i$ and the observed value z_i, and the second term Ω is the regularisation term which measures the complexity of the model and g_k corresponds to individual tree. The objective function in Equation 1.3.1 includes functions as parameters and cannot be optimized using traditional optimization methods in Euclidean space. Instead, training of XGBoost

algorithm consists of a sequential calculation process and uses K additive functions to obtain the predicted value $\hat{z_i}^{(t)}$. For each step t the predicted value is

$$\begin{aligned}
\hat{z}_i^{(0)} &= 0 \\
\hat{z}_i^{(1)} &= g_1(f_i) = \hat{z}_i^{(0)} + g_1(f_i) \\
\hat{z}_i^{(2)} &= g_1(f_i) + g_2(f_i) = \hat{z}_i^{(1)} + g_2(f_i) \\
&\dots \\
\hat{z}_i^{(t)} &= \sum_{k=1}^{t} g_k(f_i) = \hat{z}_i^{(t-1)} + g_t(f_i).
\end{aligned}$$

Substituting the prediction of the i-th instance at the t-th iteration $\hat{z}_i^{(t)}$ in the objective function

$$\mathbb{L}^{(t)} = \sum_{i=1}^{N} l(z_i, \hat{z}_i^{(t-1)} + g_t(f_i)) + \Omega(g_t). \tag{1.3.2}$$

For each iteration, g_t tree needs to be added to minimise the objective function in Equation 1.3.2. The regularization term is

$$\Omega(g) = \gamma T + \frac{1}{2}\lambda ||\omega||^2,$$

where ω represents the vector scores in the leaves, T is the number of leaves in the tree, γ and λ are the regularisation parameters.

In a typical classification problem, the extreme gradient boosting algorithm is trained to minimise a loss function with respect to classification errors. In this thesis, the classification error is not a concern, but minimising the average forecast error is. Hence, a customised loss function is used to include the information on the forecast error rather than the classification error. Extreme gradient boosting algorithm implementation allows easy changes to the objective function. It only requires the output, the gradient and the Hessian of the objective, whereas other methods require reimplementation of the majority part of the code. A detailed description of the methodology is given in Chapter 4.

1.3.3 Efficient Bayesian multivariate surface regression approach.

The methodological framework presented in Chapter 5 is based on the efficient Bayesian multivariate surface regression introduced by Li and Villani (2013). This approach has a number of advantages: i) $\mathbf{Y}$ is multivariate, which means several responses are predicted simultaneously; ii) the approach allows for interactions between features; iii) the regression splines used here are able to model the non-linear relationship between features and the response variables; and iv) compared with the other spline-based models, this approach allows the knots to move freely in the feature space, and thus a lower number of knots is usually required. A description of this approach is provided in Chapter 5. The estimation and computation details can be found in Li and Villani (2013).

1.4 Objectives and thesis outline

This is a 'thesis by publication', which consists of an introduction and conclusion with published papers in between. The main goal of the thesis is to provide support to practitioners in forecasting large collections of time series, with vectors of features computed from time series. Centralising on this main goal, the objectives and the structure of the thesis can be outlined as follows.

There are many measurable features of time series. For example, one can characterise a time series by the mean value, or by a measure of strength of seasonality, and so on. The challenge is in uncovering features that will help to select suitable models for forecasting. In response to the results of the M3 competition, Lawrence (2001) wrote:

> 'What is needed now is analysis to determine what are the specific time series characteristics for which each technique is generally best and also what are the time series characteristics for which it does not really matter which technique (or set of techniques) is chosen?'.

The **first** objective of this thesis is to identify a suitable set of features that are useful in selecting a good model for forecasting. Chapter 2 and Chapter 4 introduce a collection of time series features that are useful in selecting models for forecasting.

The **second** objective is to use a meta-learning approach with a range of features computed from the time series to recommend the way the forecasts are computed. There are many different ways the large-scale time series forecasting problem can be approached based on the meta-learning framework. Hence, this overreaching goal is divided into three constituent aims.

1. The first aim is to develop a meta-learning framework that will help identify the 'best' model for a given series. Chapter 2 is dedicated to achieving this aim. A random forest classifier is used to identify the 'best' model using time series features. The first approach treats the algorithm selection problem as a classification task. This algorithm is called FFORMS (Feature-based FORecast Model Selection). Chapter 3 extends this approach in order to handle high frequency data with multiple seasonal components.

2. The second aim is to develop a meta-learning framework to obtain weights for forecast combinations. Chapter 4 presents the second algorithm, FFORMA (Feature-based FORecast Model Averaging), in which gradient boosting is used to obtain the weights for forecast combinations. This approach achieved second place in the M4 competition (Makridakis, Spiliotis, and Assimakopoulos, 2018).

3. The third aim is to develop a meta-learning framework that allows the ranking of models according to their relative performance without calculating forecasts from all available individual models in the pool. Chapter 5 is dedicated to achieving this aim. The efficient Bayesian multivariate surface regression approach is used to estimate the forecast error for each model, and then using the predicted errors, the models are ranked to identify the 'best' individual model or 'best subset' of models to compute forecasts.

In the discussion of the M3 competition Armstrong (2001) raises the point that researchers often fail to describe under which conditions their method performs well and explain the reasons for the expectations. While the literature has concentrated mainly on the use of meta-learning for algorithm selection, not much effort has been put into identifying what is happening inside the framework. The **third** objective is to explore the relationship between

the features of time series and the choice of different models using the meta-learning frameworks introduced in this thesis. Firstly, Chapter 3 addresses this third objective. What is happening under the hood of the FFORMS framework is explored, thereby providing an understanding of what features lead to the different choices of forecast models and how different features influence the predicted outcome. This is accomplished using model-agnostic machine-learning interpretability approaches. Secondly, Chapter 5 continues addressing this third objective. The instance space defined by features is further explored to understand how certain features of time series influence model selection and also to explain how these different types of instances are located in the feature space.

The **fourth** objective is to develop a free and open-source R package for the methods introduced in this thesis. The FFORMS algorithm is implemented in the open source R package `seer`. The FFORMPP algorithm is implemented in the open source R package `fformpp`.

Chapter 6 concludes summarising the findings of the thesis, describing the software implemented and highlighting areas for future research.

Chapter 2

Meta-learning how to forecast time series

Meta-learning how to forecast time series

Thiyanga S Talagala
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: thiyanga.talagala@monash.edu
Corresponding author

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: rob.hyndman@monash.edu

George Athanasopoulos
Department of Econometrics and Business Statistics,
Monash University, VIC 3145, Australia.
Email: george.athanasopoulos@monash.edu

7 November 2019

JEL classification: C10,C14,C22

Meta-learning how to forecast time series

Abstract

A crucial task in time series forecasting is the identification of the most suitable forecasting method. We present a general framework for forecast-model selection using meta-learning. A random forest is used to identify the best forecasting method using only time series features. The framework is evaluated using time series from the M1 and M3 competitions and is shown to yield accurate forecasts comparable to several benchmarks and other commonly used automated approaches of time series forecasting. A key advantage of our proposed framework is that the time-consuming process of building a classifier is handled in advance of the forecasting task at hand.

Keywords: FFORMS (Feature-based FORecast-model Selection), Time series features, Random forest, Algorithm selection problem, Classsification

1 Introduction

Forecasting is a key activity for any business to operate efficiently. The rapid advances in computing technologies have enabled businesses to keep track of large numbers of time series. Hence, it is becoming increasingly common to have to regularly forecast many millions of time series. For example, large scale businesses may be interested in forecasting sales, cost, and demand for their thousands of products across various locations, warehouses, etc. Technology companies such as Google collect many millions of daily time series such as web-click logs, web search counts, queries, revenues, number of users for different services, etc. Such large collections of time series require fast automated procedures generating accurate forecasts. The scale of these tasks has raised some computational challenges we seek to address by proposing a new fast algorithm for model selection and time series forecasting.

Two alternative strategies for generating such a large number of forecasts are: (1) to either use a single forecasting method across all the time series; or (2) to select an appropriate forecasting method for each time series individually. It is highly unlikely that a single method

will consistently outperform judiciously chosen competitor methods across all time series. We therefore reject the former strategy and focus on an approach for selecting an individual forecasting method for each time series.

Selecting the most appropriate model for forecasting a given time series can be challenging. Two of the most commonly used automated algorithms are the exponential smoothing (`ets`) algorithm of Hyndman et al. (2002) and the ARIMA (`auto.arima`) algorithm of Hyndman & Khandakar (2008). Both algorithms are implemented in the forecast package in R (R Core Team 2018; Hyndman et al. 2018). In this paradigm, a class of models is selected in advance, and many models within that class are estimated for each time series. The model with the smallest AICc value is chosen and used for forecasting. This approach relies on the expert judgement of the forecaster in first selecting the most appropriate class of models to use, as it is not usually possible to compare AICc values between model classes due to differences in the way the likelihood is computed, and the way initial conditions are handled.

An alternative approach, which avoids selecting a class of models *a priori*, is to use a simple "hold-out" test set; but then there is often insufficient data to draw a reliable conclusion. To overcome this drawback, time series cross-validation can be used (Hyndman & Athanasopoulos 2018); then models from many different classes may be applied, and the model with the lowest cross-validated MSE selected. However, this increases the computation time involved considerably (at least to order n^2 where n is the number of series to be forecast).

Clearly, there is a need for a fast and scalable algorithm to automate the process of selecting models with the aim of forecasting. We refer to this process as forecast-model selection. We propose a general meta-learning framework using features of the time series to select the class of models, or even the specific model, to be used for forecasting. The forecast-model selection process is carried out using a classification algorithm — we use the time series features as inputs, and the "best" forecasting model as the output. The classifier is built using a large historical collection of time series, in advance of the forecasting task at hand. Hence, this is an "offline" procedure.

The "online" process of generating forecasts only involves calculating the features of a time series and using the pre-trained classifier to identify the best forecasting model. Hence, generating forecasts only involves the estimation of a single forecasting model, with no need to estimate large numbers of models within a class, or to carry out a computationally-intensive cross-validation procedure. We refer to this general framework as FFORMS

(**F**eature-based **FOR**ecast-**M**odel **S**election).

The rest of this paper is organized as follows. We review the related work in Section 2. In Section 3 we explain the detailed components and procedures of our proposed framework for forecast-model selection. In Section 4 we present the results, followed by the conclusions and future work in Section 5.

2 Literature review

2.1 Time series features

Rather than work with the time series directly at the level of individual observations, we propose analysing time series via an associated "feature space". A time series feature is any measurable characteristic of a time series. For example, Figure 1 shows the time-domain representation of six time series taken from the M3 competition (Makridakis & Hibon 2000) while Figure 2 shows a feature-based representation of the same time series. Here only two features are considered: the strength of seasonality and the strength of trend, calculated based on the measures introduced by Wang, Smith-Miles & Hyndman (2009). Time series in the lower right quadrant of Figure 2 are non-seasonal but trended, while there is only one series with both high trend and high seasonality. We also see how the degree of seasonality and trend varies between series. Other examples of time series features include autocorrelation, spectral entropy and measures of self-similarity and nonlinearity. Fulcher & Jones (2014) identified 9000 operations to extract features from time series.

The choice of the most appropriate set of features depends on both the nature of the time series being analysed, and the purpose of the analysis. In Section 4, we study the time series from the M1 and M3 competitions (Makridakis et al. 1982; Makridakis & Hibon 2000), and we select features for the purpose of forecast-model selection. The M1 and M3 competitions involve time series of differing length, scale and other properties. We include length as one of our features, but the remaining features are independent of scale and asymptotically independent of the length of the time series (i.e., they are ergodic). As our main focus is forecasting, we select features which have discriminatory power in selecting a good model for forecasting.

2.2 What makes features useful for forecast-model selection?

Reid (1972) points out that the performance of forecasting methods changes according to the nature of the data. Exploring the reasons for these variations may be useful in selecting

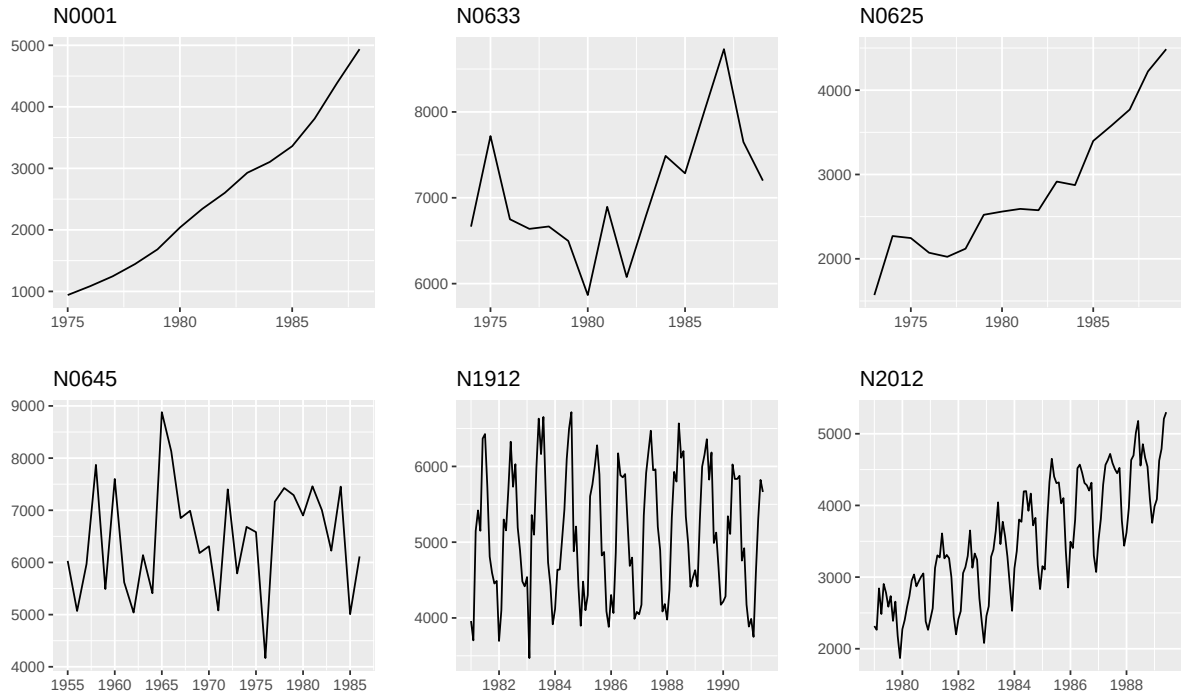

Figure 1: *Time-domain representation of time series*

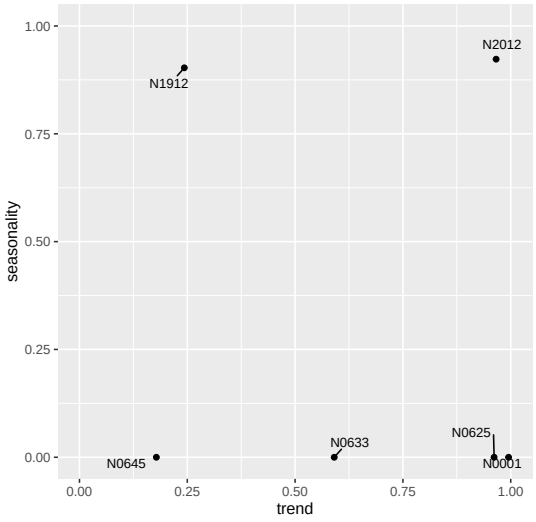

Figure 2: *Feature-based representation of time series*

the most appropriate model. In response to the results of the M3 competition (Makridakis & Hibon 2000), similar ideas have been put forward by others. Hyndman (2001), Lawrence (2001) and Armstrong (2001) argue that the characteristics of a time series may provide useful insights into which methods are most appropriate for forecasting.

Many time series forecasting techniques have been developed to capture specific characteristics of time series that are common in a particular discipline. For example, GARCH models were introduced to account for time-varying volatility in financial time series, and ETS mod-

els were introduced to handle the trend and seasonal patterns which are typical in quarterly and monthly sales data. An appropriate set of features should reveal the characteristics of the time series that are useful in determining the best forecasting method.

Several researchers have introduced rules for forecasting based on features (Collopy & Armstrong 1992; Adya et al. 2001; Wang, Smith-Miles & Hyndman 2009). Most recently Kang, Hyndman & Smith-Miles (2017) applied principal component analysis to project a large collection of time series into a two dimensional feature space in order to visualize what makes a particular forecasting method perform well or not. The features they considered were spectral entropy, first-order auto-correlation coefficient, strength of trend, strength of seasonality, seasonal period and the optimal Box-Cox transformation parameter. They also proposed a method for generating new time series based on specified features.

2.3 Meta-learning for algorithm selection

John Rice was an early and strong proponent of the idea of meta-learning, which he called the algorithm selection problem (ASP) (Rice 1976). The term *meta-learning* started to appear with the emergence of the machine-learning literature. Rice's framework for algorithm selection is shown in Figure 3 and comprises four main components. The problem space, P, represents the data sets used in the study. The feature space, F, is the range of measures that characterize the problem space P. The algorithm space, A, is a list of suitable candidate algorithms which can be used to find solutions to the problems in P. The performance metric, Y, is a measure of algorithm performance such as accuracy, speed, etc. A formal definition of the algorithm selection problem is given by Smith-Miles (2009), and repeated below.

> **Algorithm selection problem**. For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

The main challenge in ASP is to identify the selection mapping S from the feature space to the algorithm space. Even though Rice's framework articulates a conceptually rich framework, it does not specify how to obtain S. This gives rise to the meta-learning approach.

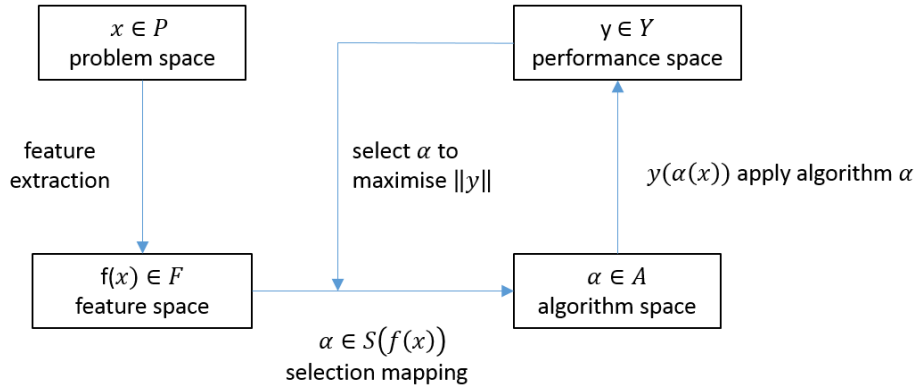

Figure 3: *Rice's framework for the Algorithm Selection Problem.*

2.4 Forecast-model selection using meta-learning

Selecting models for forecasting can be framed according to Rice's ASP framework.

> **Forecast-model selection problem**. For a given time series $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into the algorithm space A, such that the selected algorithm $\alpha \in A$ minimizes forecast accuracy error metric $y(\alpha(x)) \in Y$ on the test set of the time series.

Existing methods differ with respect to the way they define the problem space (A), the features (F), the forecasting accuracy measure (Y) and the selection mapping (S).

Collopy & Armstrong (1992) introduced 99 rules based on 18 features of time series, in order to make forecasts for economic and demographic time series. This work was extended by Armstrong (2001) to reduce human intervention.

Shah (1997) used the following features to classify time series: the number of observations, the ratio of the number of turning points to the length of the series, the ratio of number of step changes, skewness, kurtosis, the coefficient of variation, autocorrelations at lags 1–4, and partial autocorrelations at lag 2–4. Casting Shah's work in Rice's framework, we can specify: $P = 203$ quarterly series from the M1 competition (Makridakis et al. 1982); $A = 3$ forecasting methods, namely simple exponential smoothing, Holt-Winters exponential smoothing with multiplicative seasonality, and a basic structural time series model; $Y =$ mean squared error for a hold-out sample. Shah (1997) learned the mapping S using discriminant analysis.

Prudêncio & Ludermir (2004) was the first paper to use the term "meta-learning" in the context of time series model selection. They studied the applicability of meta-learning

approaches for forecast-model selection based on two case studies. Again using the notation above, we can describe their first case study with: A contained only two forecasting methods, simple exponential smoothing and a time-delay neural network; Y = mean absolute error; F consisted of 14 features, namely length, autocorrelation coefficients, coefficient of variation, skewness, kurtosis, and a test of turning points to measure the randomness of the time series; S was learned using the C4.5 decision tree algorithm. For their second study, the algorithm space included a random walk, Holt's linear exponential smoothing and AR models; the problem space P contained the yearly series from the M3 competition (Makridakis & Hibon 2000); F included a subset of features from the first study; and Y was a ranking based on error. Beyond the task of forecast-model selection, they used the NOEMON approach to rank the algorithms (Kalousis & Theoharis 1999).

Lemke & Gabrys (2010) studied the applicability of different meta-learning approaches for time series forecasting. Their algorithm space A contained ARIMA models, exponential smoothing models and a neural network model. In addition to statistical measures such as the standard deviation of the de-trended series, skewness, kurtosis, length, strength of trend, Durbin-Watson statistics of regression residuals, the number of turning points, step changes, a predictability measure, nonlinearity, the largest Lyapunov exponent, and auto-correlation and partial-autocorrelation coefficients, they also used frequency domain based features. The feed forward neural network, decision tree and support vector machine approaches were considered to learn the mapping S.

Wang, Smith-Miles & Hyndman (2009) used a meta-learning framework to provide recommendations as to which forecast method to use to generate forecasts. In order to evaluate forecast accuracy, they introduced a new measure Y = *simple percentage better (SPB)*, which calculates the forecasting accuracy of a method against the forecasting accuracy error of random walk model. They used a feature set F comprising nine features: strength of trend, strength of seasonality, serial correlation, nonlinearity, skewness, kurtosis, self-similarity, chaos and periodicity. The algorithm space A included eight forecast-models, namely, exponential smoothing, ARIMA, neural networks and random walk model; while the mapping S was learned using the C4.5 algorithm for building decision trees. In addition, they used SOM clustering on the features of the time series in order to understand the nature of time series in a two-dimensional setting.

The set of features introduced by Wang, Smith-Miles & Hyndman (2009) was later used by Widodo & Budi (2013) to develop a meta-learning framework for forecast-model selection.

The authors further reduced the dimensionality of time series by performing principal component analysis on the features.

More recently, Kück, Crone & Freitag (2016) proposed a meta-learning framework based on neural networks for forecast-model selection. Here, $P = 78$ time series from the NN3 competition were used to build the meta-learner. They introduced a new set of features based on forecasting errors. The average symmetric mean absolute percentage error was used to identify the best forecast-models for each series. They classify their forecast-models in the algorithm space A, comprising single, seasonal, seasonal-trend and trend exponential smoothing. The mapping S was learned using a feed-forward neural network. Further, they evaluated the performance of different sets of features for forecast-model selection.

3 Methodology

Our proposed FFORMS framework, presented in Figure 4, builds on this preceding research. The offline and online phases are shown in blue and red respectively. A classification algorithm (the meta-learner) is trained during the offline phase and is then used to select an appropriate forecast model for a new time series in the online phase.

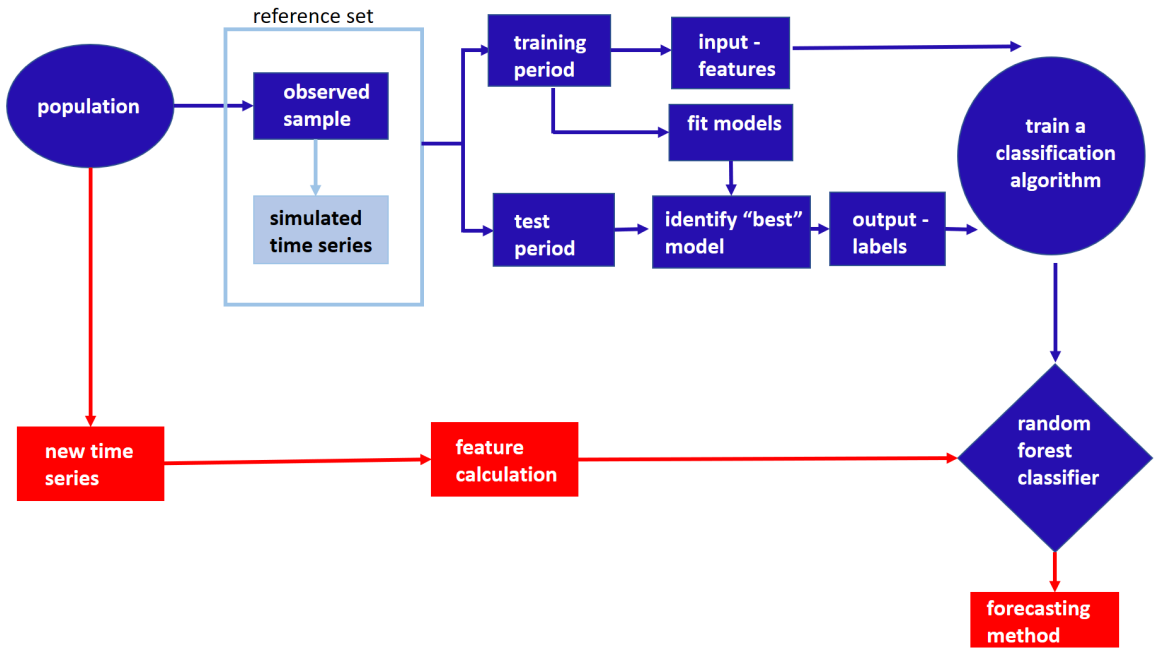

Figure 4: *FFORMS (Feature-based FORecast-Model Selection) framework. The offline phase is shown in blue and the online phase in red.*

In order to train our classification algorithm, we need a large collection of time series which are similar to those we will be forecasting. We assume that we have an essentially infinite

population of time series, and we take a sample of them in order to train the classification algorithm denoted as the “observed sample”. The new time series we wish to forecast can be thought of as additional draws from the same population. Hence, any conclusions made from the classification framework refer only to the population from which the sample has been selected. We may call this the “target population” of time series. It is important to have a well-defined target population to avoid misapplying the classification rules. In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population (we provide details and discussion in Section 3.1 that follows). We denote the total collection of time series used for training the classifier as the “reference set”. Each time series within the reference set is split into a training period and a test period. From each training period we compute a range of time series features, and fit a selection of candidate models. The calculated features form the input vector to the classification algorithm. Using the fitted models, we generate forecasts and identify the “best” model for each time series based on a forecast error measure (e.g., MASE) calculated over the test period. The models deemed “best” form the output labels for the classification algorithm. The pseudo code for our proposed framework is presented in Algorithm 1 below. In the following sections, we briefly discuss aspects of the offline phase.

3.1 Augmenting the observed sample with simulated time series

In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population. This process may be useful when the number of observed time series is too small to build a reliable classifier. Alternatively, we may wish to add more of some particular types of time series to the reference set in order to get a more balanced sample for the classification. In order to produce simulated series that are similar to those in the population, we consider two classes of data generating processes: exponential smoothing models and ARIMA models. Using the automated `ets` and `auto.arima` algorithms (Hyndman et al. 2018) we identify models, based on model selection criteria (such as AICc) and simulate multiple time series from the selected models within each model class. Assuming the models produce data that are similar to the observed time series ensures that the simulated series are similar to those in the population. As this is done in the offline phase of the algorithm the computational time in producing these simulated series is of no real consequence.

Algorithm 1 The FFORMS framework - Forecasting based on meta-learning.

Offline phase - train the classifier

Given:

$O = \{x_1, x_2, \ldots, x_n\}$: the collection of n observed time series.
L : the set of class labels (e.g. ARIMA, ETS, SNAIVE).
F : the set of functions to calculate time series features.
$nsim$: number of series to be simulated.
B : number of trees in the random forest.
k : number of features to be selected at each node.

Output:

FFORMS classifier

Prepare the reference set

For $i = 1$ to n:

1: Fit ARIMA and ETS models to x_i.
2: Simulate $nsim$ time series from each model in step 1.
3: The time series in O and simulated time series in step 2 form the reference set $R = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_N\}$ where $N = n + nsim$.

Prepare the meta-data

For $j = 1$ to N:

4: Split x_j into a training period and test period.
5: Calculate features F based on the training period.
6: Fit L models to the training period.
7: Calculate forecasts for the test period from each model.
8: Calculate forecast error measure over the test period for all models in L.
9: Select the model with the minimum forecast error.
10: Meta-data: input features (step 5), output labels (step 9).

Train a random forest classifier

11: Train a random forest classifier based on the meta-data.
12: Random forest: the ensemble of trees $\{T_b\}_1^B$.

Online phase - forecast a new time series

Given:

FFORMS classifier from step 12 .

Output:

class labels for new time series x_{new}.

13: For x_{new} calculate features F.
14: Let $\hat{C}_b(x_{new})$ be the class prediction of the b^{th} random forest tree. Then class label for x_{new} is $\hat{C_{rf}}(x_{new}) = majorityvote\{\hat{C}_b(x_{new})\}_1^B$.

3.2 Input: features

Our proposed FFORMS algorithm requires features that enable identification of a suitable forecast model for a given time series. Therefore, the features used should capture the dynamic structure of the time series, such as trend, seasonality, autocorrelation, nonlinearity, heterogeneity, and so on. Furthermore, interpretability, robustness to outliers, scale and

length independence should also be considered when selecting features for this classification problem. A comprehensive description of the features used in the experiment is presented in Table 2.

A feature of the proposed FFORMS framework is that its online phase is fast compared to the time required for implementing a typical model selection or cross-validation procedure. Therefore, we consider only features that can be computed rapidly, as this computation must be done during the online phase.

3.3 Output: labels

The task of the classification algorithm is to identify the "best" forecasting method for a given time series. It is not possible to train the classifier for all possible classes of models. However, we should consider enough possibilities so that the algorithm can be used for forecast-model selection with some confidence. The candidate models considered as labels will depend on the observed time series. For example, if we have only non-seasonal time series, and no chaotic features, we may wish to restrict the models to white noise, random walk, ARIMA and ETS processes. Even in this simple scenario, the number of possible models can be quite large.

Each candidate model considered is estimated strictly on the training period of each series in the reference set. Forecasts are then generated for the test period over which the chosen forecast accuracy measure is calculated. The model with the lowest forecast error measure over the test period is deemed "best".

This step is the most computationally intensive and time-consuming, as each candidate model has to be applied on each series in the reference set. However, as this task is done during the offline phase of the FFORMS framework, the time involved and computational cost associated are of no real significance and are completely controlled by the user. The more candidate models that are considered as labels, the more computational time is required; however the pay-off could be significant gains in forecast accuracy. This is similar to other computationally intensive supervised learning methods such as deep learning. The training process could take couple of days, but the application could be done within few seconds even with your phone.

3.4 Random forest algorithm

A random forest (Breiman 2001) is an ensemble learning method that combines a large number of decision trees using a two-step randomization process. Let $(\boldsymbol{f}_1, z_1), (\boldsymbol{f}_2, z_2), \ldots, (\boldsymbol{f}_N, z_N)$

represent the reference set, where input $\boldsymbol{f}_i$ is an m-vector of features, output z_i corresponds to the class label of the ith time series, and N is the number of series in the reference set. Each tree in the forest is grown based on a bootstrap sample of size N from the reference set. At each node of the tree, randomly select $k < m$ features from the full set of features. The best split is selected among those k features. The split which results in the most homogeneous subnodes is considered the best split. Various measures have been introduced to evaluate the homogeneity of subnodes, such as classification error rate, the Gini index and cross entropy (Friedman, Hastie & Tibshirani 2009). In this study, we use the Gini index to evaluate the homogeneity of a particular split. The trees are grown to the largest extent possible without pruning. To determine the class label for a new instance, features are calculated and passed down the trees. Then each tree gives a prediction and the majority vote over all individual trees leads to the final decision. In this work, we have used the randomForest package (Liaw & Wiener 2002; Breiman et al. 2018) which implements the Fortran code for random forest classification by Breiman & Cutler (2004).

4 Application to the M competition data

To test how well our proposed framework can identify suitable models for forecasting, we use the yearly, quarterly and monthly series from the M1 (Makridakis et al. 1982) and M3 competitions (Makridakis & Hibon 2000). The R package Mcomp (Hyndman 2018) provides the data for both these. We run two experiments. In the first experiment, we treat the time series from the M1 competition as the *observed sample* and the time series from the M3 competition as the *new time series* to be forecast. We reverse the sets in the second experiment, treating the M3 data as the *observed sample* and the M1 data as the *new time series*. The experimental set-ups are summarised in Table 1. These allow us to compare our results with published forecast accuracy results for each of these competitions.

Table 1: *The number of series and their sources used in the two forecast experiments.*

	Experiment 1				Experiment 2			
	Source	Yearly	Quarterly	Monthly	Source	Yearly	Quarterly	Monthly
Observed series	M1	181	203	617	M3	645	756	1428
New series	M3	645	756	1428	M1	181	203	617

In each experiment, we fit ARIMA and ETS models to the full length of each series in the corresponding *observed samples* using the `auto.arima` and `ets` functions in the forecast package. From each model fitted to the annual and quarterly data, we simulate a further 1000 series for augmenting the reference set. For the monthly time series, we simulate a

further 100 series (in order to save time in the offline calculation process). The lengths of the simulated series are set to be equal to the lengths of the series on which these are based on.

As shown in Figure 4, the task of constructing the meta-database contains two main components: (i) identification of the candidate forecast models as output labels and (ii) computation of features.

4.1 Identifying output labels

The candidate models we consider as class labels for the annual series are:

(a) White noise (WN)
(b) AR/MA/ARMA
(c) ARIMA;
(d) Random walk with drift (RWD);
(e) Random walk (RW);
(f) Theta;
(g) Exponential Smoothing Model (ETS) without trend and seasonal components;
(h) ETS with trend component and without seasonal component;
(i) ETS with damped trend component and without seasonal component;

In addition to the above nine labels, we further include the following six class labels for both quarterly and monthly data:

(j) STL-AR;
(k) ETS with trend and seasonal components
(l) ETS with damped trend and seasonal components
(m) ETS with seasonal components and without trend component
(n) SARIMA
(o) Seasonal naive method.

Most of these are self-explanatory labels based on models implemented in the forecast package using the default settings.

STL-AR refers to forecasts based on an STL decomposition applied to the time series, then an AR model is used to forecast the seasonally adjusted data, while the seasonal naive method is used to forecast the seasonal component. The two sets of forecasts are then combined to provide forecasts on the original time scale (Hyndman & Athanasopoulos 2018).

For each series in the reference set, all candidate models are estimated using the training period, and forecasts are generated for the whole of the test period (based on "a fixed origin" evaluation) as set by the competitions. The model corresponding to the smallest MASE (Hyndman & Koehler 2006) calculated over the test period is selected as the “best model” and forms the *output label* for that series.

4.2 Feature computation process

We use a set of 25 features for yearly data and a set of 30 features for seasonal data. Some of these are taken from previous studies (Wang, Smith-Miles & Hyndman 2009; Hyndman, Wang & Laptev 2015; Kang, Hyndman & Smith-Miles 2017), and we have also added new features that we believe provide some useful information. Each feature can be computed rapidly, thus making the online phase of our algorithm extremely fast. The features are summarized in Table 2, and fully described in the Appendix.

Correlation matrices for all the features calculated from the reference sets of each of the experiments are presented in Figure 5. The variability in the correlations reflects the diversity of the selected features. In other words, the features we have employed seem to capture different characteristics of the time series. Furthermore, the structure of the correlation matrices across the same frequencies of the two experiments seem to be fairly similar. This sends a strong signal that the M1 and M3 collections of time series may have similar feature spaces.

4.3 Model calibration

This is a multi-class classification task, with predictors = features, and outcome = categorical variable with all possible forecast models as its class label. In particular, an instance is a tuple $(f_{i1}, f_{i2}, \ldots, f_{im}, z_i)$, where predictors = $(f_{i1}, f_{i2}, \ldots, f_{im})$ is a vector of m features which is the input to our algorithm, and outcome is z_i the "best" forecast model. An example of instance is $(0.51, 0.12, \ldots, 3.1, SARIMA)$. The random forest (RF) algorithm is highly sensitive to class imbalance (Breiman 2001), and our reference set is unbalanced: some classes contain significantly more cases than other classes. The degree of class imbalance is reduced to some extent by augmenting the observed sample with the simulated time series. We consider three approaches to address the class imbalance in the data: (1) incorporating class priors into the RF classifier; (2) using the balanced RF algorithm introduced by Chen, Liaw & Breiman (2004); and (3) re-balancing the reference set with down-sampling. In down-sampling, the size of the reference set is reduced by down-sampling the larger classes so that they match the smallest class in size; this potentially discards some useful information.

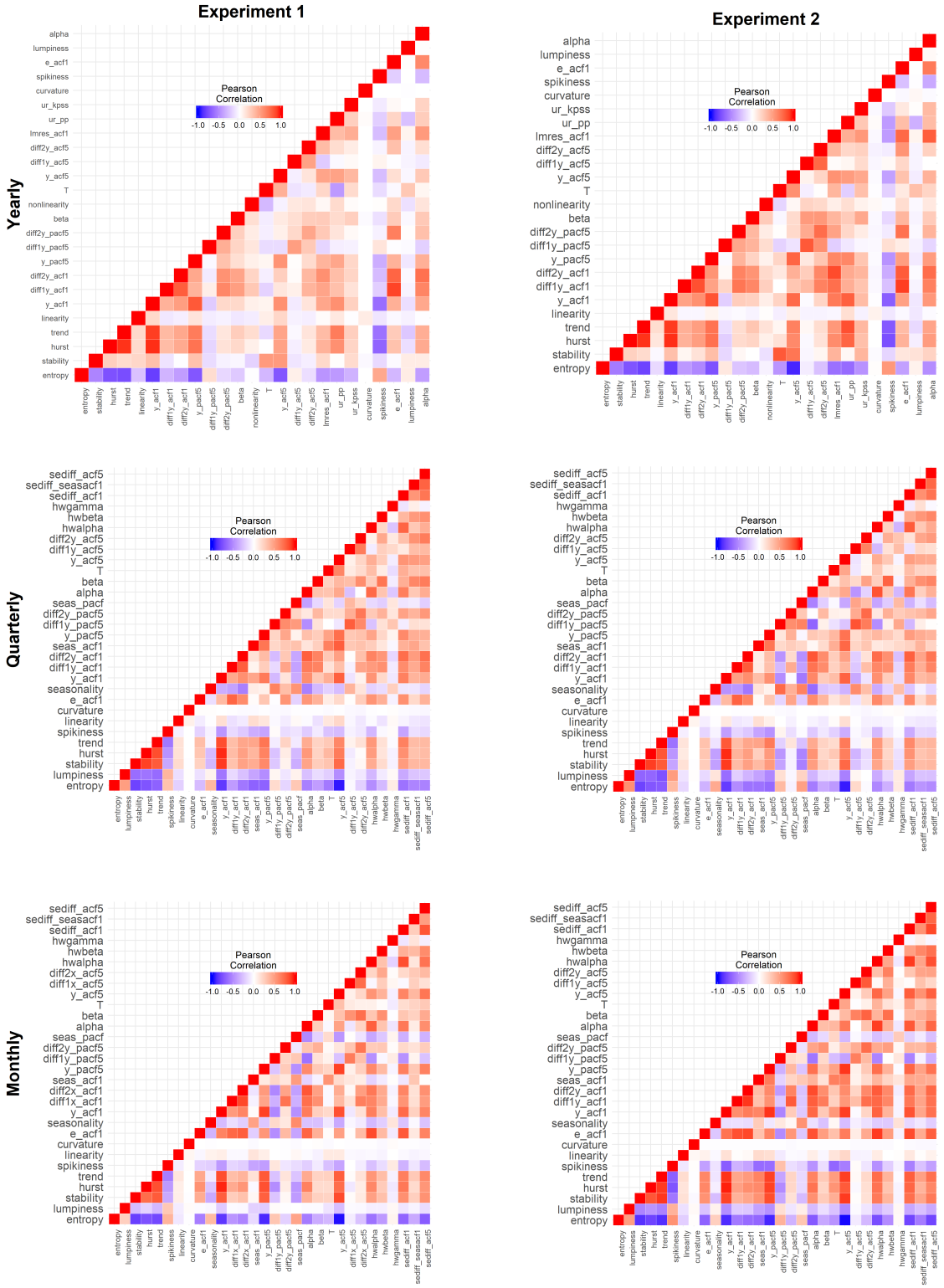

Figure 5: *Correlation matrix plots for the reference sets.*

Table 2: *Features used for selecting a forecast-model (see Appendix A for details).*

	Feature	Description	Non-seasonal	Seasonal
1	T	length of time series	✓	✓
2	trend	strength of trend	✓	✓
3	seasonality	strength of seasonality	-	✓
4	linearity	linearity	✓	✓
5	curvature	curvature	✓	✓
6	spikiness	spikiness	✓	✓
7	e_acf1	first ACF value of remainder series	✓	✓
8	stability	stability	✓	✓
9	lumpiness	lumpiness	✓	✓
10	entropy	spectral entropy	✓	✓
11	hurst	Hurst exponent	✓	✓
12	nonlinearity	nonlinearity	✓	✓
13	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓
14	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓
15	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓
16	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓
17	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓
18	ur_pp	test statistic based on Phillips-Perron test	✓	-
19	ur_kpss	test statistic based on KPSS test	✓	-
20	y_acf1	first ACF value of the original series	✓	✓
21	diff1y_acf1	first ACF value of the differenced series	✓	✓
22	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓
23	y_acf5	sum of squares of first 5 ACF values of original series	✓	✓
24	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓
25	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓
26	seas_acf1	autocorrelation coefficient at first seasonal lag	-	✓
27	sediff_acf1	first ACF value of seasonally-differenced series	-	✓
28	sediff_seasacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓
29	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓
30	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	✓
31	lmres_acf1	first ACF value of residual series of linear trend model	✓	-
32	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓
33	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓
34	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓

Comparing the results, the balanced RF algorithm and RF with down-sampling did not yield satisfactory results. We therefore only report the results obtained by the RF built on unbalanced data (RF-unbalanced) and the RF with class priors (RF-class priors). The RF algorithms are implemented by the randomForest R package (Liaw & Wiener 2002; Breiman et al. 2018). The class priors are introduced through the option `classwt`. We use the reciprocal of class size as the class priors. The number of trees `ntree` is set to 1000, and the number of randomly selected features `k` is set to be one third of the total number of features available.

Our aim is different from most classification problems in that we are not interested in accurately predicting the class, but in finding the best possible forecast model. It is possible that two models produce almost equally accurate forecasts, and therefore it is not important

whether the classifier picks one model over the other. Therefore we report the forecast accuracy obtained from the FFORMS framework, rather than the classification accuracy.

4.4 Summary of the main results

We build separate RF classifiers for yearly data, quarterly data and monthly data. For the second experiment (for which data from the M3 competition are the observed series), in the case of yearly and quarterly data we take a subset of the simulated time series when training the RF-unbalanced and RF-class priors, to reduce the size of the reference set (these are shown in yellow in the figures that follow). The subsets are selected randomly according to the proportions of output labels in the observed samples. This ensures that our reference set shares similar characteristics to the observed sample.

We use principal component analysis to visualize the feature-spaces of the different time series collections. We compute the principal components projection using the features in the observed sample, and then project the simulated time series and the new time series using the same projection. The results are shown in Figures 6–7, where the first three principal components are plotted against each other. Figure 6 refers to Experiment 1 and Figure 7 to Experiment 2. The points on each plot represent individual time series. In each figure the first column of plots refers to the yearly data, the middle column to the quarterly data and the last column to the monthly data. The plots show that the first three principal components explain 62.5%, 62.4% and 58.9% of the total variation in the yearly, quarterly and monthly M1 data and 62.2% 64.7% and 66.0% of the total variation in the yearly, quarterly and monthly M3 data.

The plots show that the distribution of the observed time series over the PCA space (represented by the black dots) is very similar to that of the new time series (represented by the orange dots). More importantly, we see that the distribution of the simulated time series (represented by the green dots and yellow dots - the yellow dots are a subset of the simulated series as mentioned above) clearly nests and fills in the space of the new time series. Further, in both experiments, all the *observed time series* fall within the space of all simulated data. This strongly indicates that we have not reduced the feature diversity from the observed sample. By augmenting the observed series with simulated time series, we have been able to increase the diversity and evenness of the feature space in the reference sets.

We compare the accuracy of the forecasts generated from the FFORMS framework to the following commonly-used forecasting methods:

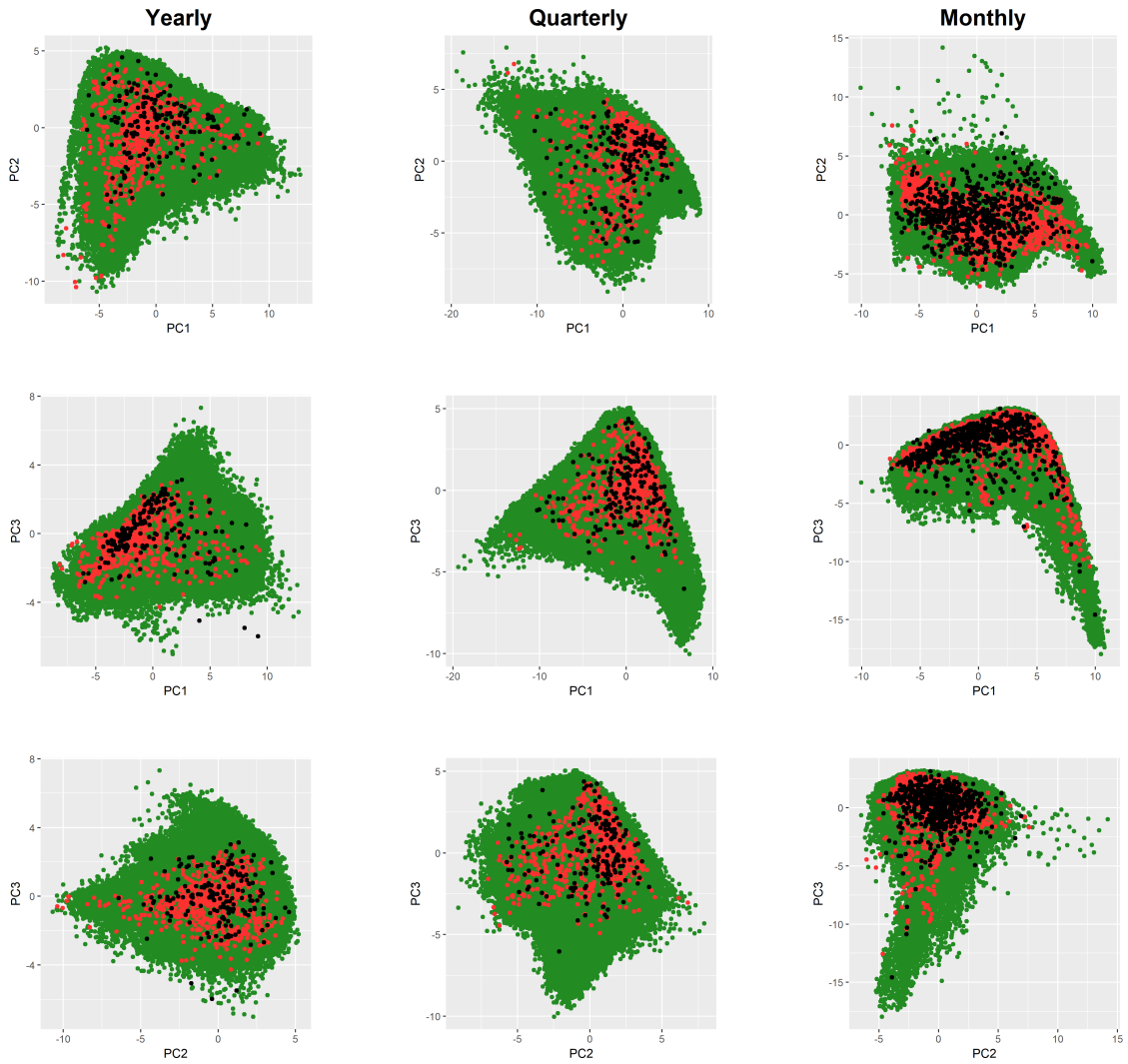

Figure 6: *Experiment 1: Distribution of time series in the PCA space. Distribution of yearly series are shown in the first column, distribution of quarterly series are shown in the second column, and the distribution of monthly series are shown in the third column. On each graph, green indicates simulated time series, black indicates observed time series, while orange denotes new time series.*

1. automated ARIMA algorithm of Hyndman & Khandakar (2008);
2. automated ETS algorithm of Hyndman & Khandakar (2008);
3. Random walk with drift (RWD);
4. Random walk model (RW);
5. White noise process (WN);
6. Theta method;
7. STL-AR method (for seasonal data);
8. seasonal naive (for seasonal data).

The automated ARIMA and ETS algorithms are implemented using the `auto.arima` and `ets` functions available in the forecast package in R. Each method is implemented on the

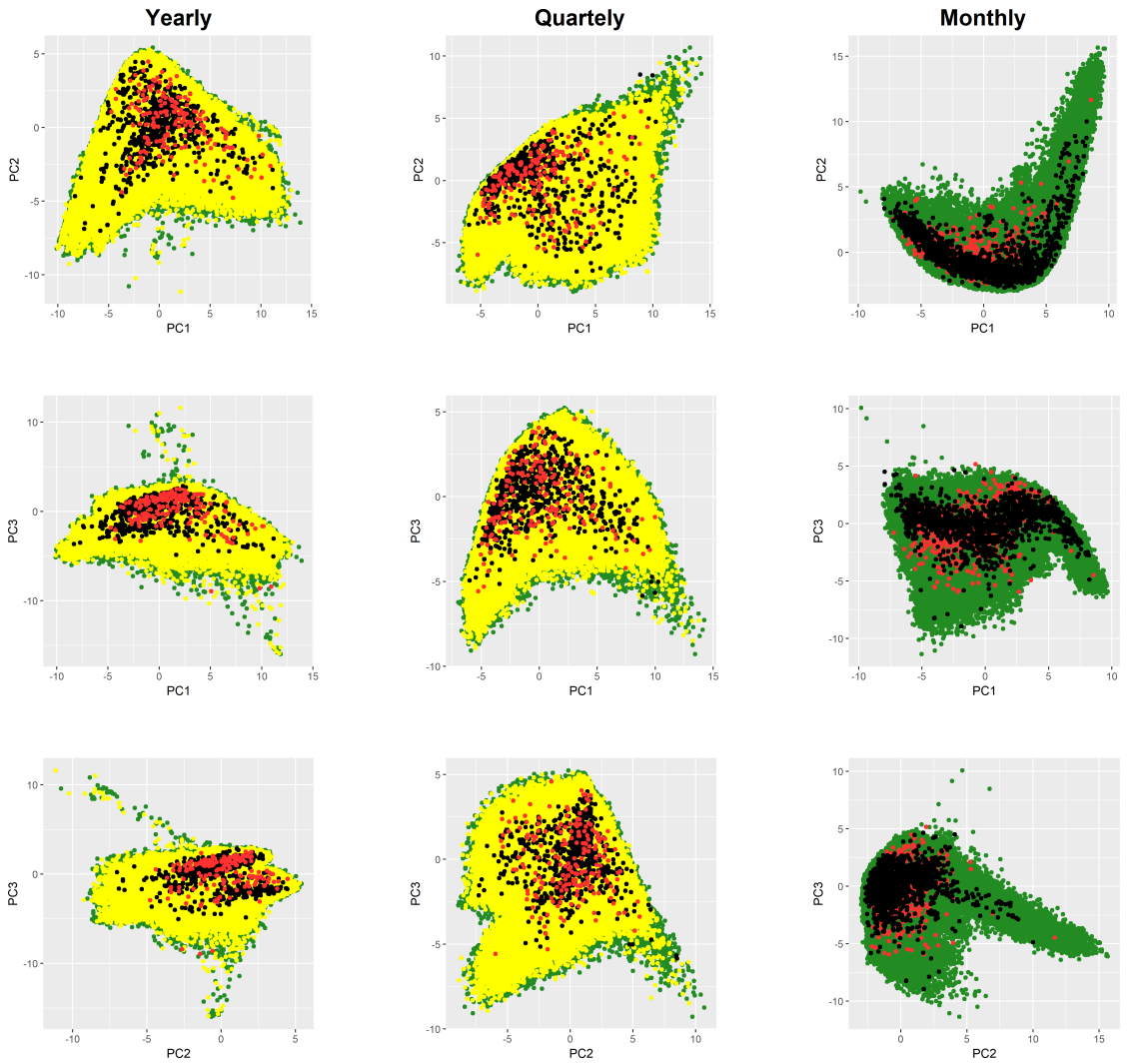

Figure 7: *Experiment 2: Distribution of time series in the PCA space. Distribution of yearly series are shown in the first column, distribution of quarterly series are shown in the second column, and the distribution of monthly series are shown in the third column. On each graph, green indicates simulated time series, yellow shows a subset of simulated time series, black indicates observed time series, while orange denotes new time series.*

training period of the new series and forecasts are computed for the full length of the test period. This follows closely the forecast evaluation process implemented in the M1 and M3 forecasting competitions. The MASE is computed for each forecast horizon, by averaging the absolute scaled forecast errors across all series. The rank is calculated by averaging the ranks of each competing method across all forecast horizons and series. The results are presented in Table 3. The associated box-and-whisker plots are shown in Figures 8 through 10 in Appendix B. The lowest MASE and average rank values corresponding to the best performing method are presented in bold font.

In general, our proposed FFORMS meta-learning algorithm performs quite well in both experimental settings. It consistently ranks in the top most accurate forecasting methods

Table 3: *MASE values calculated over the new series for Experiments 1 and 2.*

	Experiment 1: new series M3					Experiment 2: new series M1				
	Yearly					Yearly				
	$h=1$	$1-2$	$1-4$	$1-6$	Rank	$h=1$	$1-2$	$1-4$	$1-6$	Rank
RF-unbalanced	1.06	1.40	2.17	2.82	3.50	**0.98**	**1.40**	**2.43**	3.39	**1.50**
RF-class priors	1.04	1.38	2.15	2.79	2.50	1.01	**1.40**	**2.43**	**3.38**	**1.50**
auto.arima	1.11	1.48	2.28	2.96	5.83	1.06	1.47	2.51	3.47	3.33
ets	1.09	1.44	2.20	2.86	4.67	1.12	1.59	2.72	3.77	5.00
WN	6.54	6.91	7.48	8.07	9.00	6.38	7.08	8.59	10.01	8.00
RW	1.24	1.68	2.48	3.17	8.00	1.35	2.00	3.50	4.89	7.00
RWD	**1.03**	**1.36**	**2.05**	**2.63**	**1.00**	1.04	1.44	2.51	3.49	3.67
Theta	1.12	1.47	2.18	2.77	3.50	1.15	1.70	3.00	4.19	6.00
	Quarterly					Quarterly				
	$h=1$	$1-4$	$1-6$	$1-8$	Rank	$h=1$	$1-4$	$1-6$	$1-8$	Rank
RF-unbalanced	0.59	**0.81**	**0.97**	1.12	**2.25**	**0.74**	**1.08**	**1.35**	**1.57**	**1.00**
RF-class priors	0.59	0.82	**0.97**	1.13	3.13	0.76	1.12	1.40	1.62	2.63
auto.arima	0.59	0.85	1.02	1.19	4.75	0.78	1.17	1.50	1.74	5.25
ets	**0.56**	0.82	0.99	1.17	3.75	0.78	1.11	1.42	1.66	3.00
WN	3.25	3.59	3.70	3.87	10.00	3.97	4.27	4.45	4.64	10.00
RW	1.14	1.16	1.32	1.46	7.00	0.97	1.35	1.67	1.95	7.50
RWD	1.20	1.17	1.36	1.47	6.50	0.95	1.26	1.56	1.81	5.38
STL-AR	0.70	1.27	1.60	1.91	8.34	0.96	1.63	2.05	2.43	8.63
Theta	0.62	0.83	**0.97**	**1.11**	2.50	0.79	1.13	1.42	1.67	3.88
Snaive	1.11	1.09	1.30	1.43	6.75	1.52	1.56	1.87	2.08	7.75
	Monthly					Monthly				
	$h=1$	$1-6$	$1-12$	$1-18$	Rank	$h=1$	$1-6$	$1-12$	$1-18$	Rank
RF-unbalanced	0.60	0.68	0.76	0.87	3.22	0.61	0.76	**0.90**	**1.03**	**1.77**
RF-class priors	0.60	0.67	0.75	**0.86**	**2.00**	0.60	**0.75**	0.92	1.06	2.83
auto.arima	**0.55**	**0.64**	**0.74**	0.87	2.83	0.60	0.76	0.96	1.12	4.94
ets	**0.55**	**0.64**	**0.74**	**0.86**	2.72	**0.59**	0.76	0.93	1.07	3.44
WN	2.01	2.08	2.15	2.27	10.00	1.93	2.09	2.18	2.28	10.00
RW	0.84	0.97	1.04	1.17	8.03	1.05	1.24	1.33	1.47	7.25
RWD	0.84	0.96	1.02	1.14	6.89	1.06	1.27	1.39	1.55	8.61
STL-AR	0.64	0.81	1.04	1.27	7.89	0.63	0.91	1.17	1.39	7.38
Theta	0.58	0.67	0.77	0.89	4.22	0.61	**0.75**	0.92	1.04	2.27
Snaive	0.95	0.97	0.99	1.15	7.19	1.06	1.11	1.14	1.31	6.47

Note: h is the length of the forecast horizon.

for forecasting both the M1 and the M3 series and most often ranks as the most accurate method. For yearly data, FFORMS ranks best for forecasting the M1 data in experiment 2 and second (to the random walk with drift) in forecasting the M3 data. For quarterly data, FFORMS ranks as the most accurate in both experiments. For monthly data, FFORMS seems to perform best for the longer horizons in both experimental settings. It also seems to be competitive for the shorter horizons with the three methods (`auto.arima`, `ets` and theta) that perform best. Comparing the results across the two experimental settings, it seems that using the FFORMS algorithm achieves marginally better results in experiment 2. Hence, this indicates that the meta-learning algorithm benefits from being trained on the larger observed sample of time series (M3 series) while forecasting a smaller new set (M1 series).

5 Discussion and conclusions

In this paper we have proposed a novel framework for forecast-model selection using meta-learning based on time series features. Our proposed FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify the best forecasting method for a new series. We have shown that the method almost always performs better than common benchmark methods, and in many cases better than the best-performing methods from both the M1 and the M3 forecasting competitions. Although we have illustrated the method using the M1 and M3 competition data, the framework is general and can be applied to any large collection of time series.

A major advantage of the FFORMS framework is that the classifier is trained offline and selecting a forecasting model for a new time series is as fast as calculating a set of features and passing these to the pretrained classifier. Therefore, unlike traditional model selection strategies or cross-validation processes, our proposed framework can be used to accurately forecast very large collections of time series requiring almost instant forecasts.

In addition to our new FFORMS framework, we have also introduced a simple set of time series features that are useful in identifying the “best” forecast method for a given time series, and can be computed rapidly. We will leave to a later study an analysis of which of these features are the most useful, and how our feature set could be reduced further without the loss of forecast accuracy.

For real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on the FFORMS random forest classifier, and the calculation of the forecasts from the chosen model. None of these steps involve substantial computation, and they can be easily parallelised when forecasting for a large number of new time series. For future work, we will explore the use of other classification algorithms within the FFORMS algorithm, and test our approach on several other large collections of time series.

Appendix A: Definition of features

Length of time series

The appropriate forecast method for a time series depends on how many observations are available. For example, shorter series tend to need simple models such as a random walk. On the other hand, for longer time series, we have enough information to be able to estimate

a number of parameters. For even longer series (over 200 observations), models with time-varying parameters give good forecasts as they help to capture the changes of the model over time.

Features based on an STL-decomposition

The strength of trend, strength of seasonality, linearity, curvature, spikiness and first autocorrelation coefficient of the remainder series, are calculated based on a decomposition of the time series. Suppose we denote our time series as $y_1, y_2, \dots, y_T$. First, an automated Box-Cox transformation (Guerrero 1993) is applied to the time series in order to stabilize the variance and to make the seasonal effect additive. The transformed series is denoted by y_t^*. For quarterly and monthly data, this is decomposed using STL (Cleveland, Cleveland & Terpenning 1990) to give $y_t^* = T_t + S_t + R_t$, where T_t denotes the trend, S_t denotes the seasonal component, and R_t is the remainder component. For non-seasonal data, Friedman's super smoother (Friedman 1984) is used to decompose $y_t^* = T_t + R_t$, and $S_t = 0$ for all t. The de-trended series is $y_t^* - T_t = S_t + R_t$, the deseasonalized series is $y_t^* - S_t = T_t + R_t$..

The strength of trend is measured by comparing the variance of the deseasonalized series and the remainder series (Wang, Smith-Miles & Hyndman 2009):

$$\text{Trend} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(T_t + R_t)\right].$$

Similarly, the strength of seasonality is computed as

$$\text{Seasonality} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(S_t + R_t)\right].$$

The linearity and curvature features are based on the coefficients of an orthogonal quadratic regression

$$T_t = \beta_0 + \beta_1\phi_1(t) + \beta_2\phi_2(t) + \varepsilon_t,$$

where $t = 1, 2, \dots, T$, and ϕ_1 and ϕ_2 are orthogonal polynomials of orders 1 and 2. The estimated value of β_1 is used as a measure of linearity while the estimated value of β_2 is considered as a measure of curvature. These features were used by Hyndman, Wang & Laptev (2015). The linearity and curvature depend on the the scale of the time series. Therefore, the time series are scaled to mean zero and variance one before these two features are computed.

The spikiness feature is useful when a time series is affected by occasional outliers. Hyndman,

Wang & Laptev (2015) introduced an index to measure spikiness, computed as the variance of the leave-one-out variances of r_t.

We compute the first autocorrelation coefficient of the remainder series, r_t.

Stability and lumpiness

The features "stability" and "lumpiness" are calculated based on tiled windows (i.e., they do not overlap). For each window, the sample mean and variance are calculated. The stability feature is calculated as the variance of the means, while lumpiness is the variance of the variances. These were first used by Hyndman, Wang & Laptev (2015).

Spectral entropy of a time series

Spectral entropy is based on information theory, and can be used as a measure of the forecastability of a time series. Series that are easy to forecast should have a small spectral entropy value, while very noisy series will have a large spectral entropy. We use the measure introduced by Goerg (2013) to estimate the spectral entropy. It estimates the Shannon entropy of the spectral density of the normalized spectral density, given by

$$H_s(y_t) := -\int_{-\pi}^{\pi} \hat{f}_y(\lambda) \log \hat{f}_y(\lambda) d\lambda,$$

where $\hat{f}_y$ denotes the estimate of the spectral density introduced by Nuttall & Carter (1982). The R package ForeCA (Goerg 2016) was used to compute this measure.

Hurst exponent

The Hurst exponent measures the long-term memory of a time series. The Hurst exponent is given by $H = d + 0.5$, where d is the fractal dimension obtained by estimating a ARFIMA$(0, d, 0)$ model. We compute this using the maximum likelihood method (Haslett & Raftery 1989) as implemented in the fracdiff package (Fraley 2012). This measure was also used in Wang, Smith-Miles & Hyndman (2009).

Nonlinearity

To measure the degree of nonlinearity of the time series, we use statistic computed in Terasvirta's neural network test for nonlinearity (Teräsvirta, Lin & Granger 1993), also used in Wang, Smith-Miles & Hyndman (2009). This takes large values when the series is nonlinear, and values around 0 when the series is linear.

Parameter estimates of an ETS model

The ETS(A,A,N) model (Hyndman et al. 2008) produces equivalent forecasts to Holt's linear trend method, and can be expressed as follows:

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t,
\end{aligned}
$$

where α is the smoothing parameter for the level, and β is the smoothing parameter for the trend. We include the parameter estimates of both α and β in our feature set for yearly time series. These indicate the variability in the level and slope of the time series.

The ETS(A,A,A) model (Hyndman et al. 2008) produces equivalent forecasts to Holt-Winters' additive method, and can be expressed as follows:

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t, \\
s_t &= s_{t-m} + \gamma\varepsilon_t,
\end{aligned}
$$

where γ is the smoothing parameter for the seasonal component, and the other parameters are as above. We include the parameter estimates of α, β and γ in our feature set for monthly and quarterly time series. The value of γ provides a measure for the variability of the seasonality of a time series.

Unit root test statistics

The Phillips-Perron test is based on the regression $y_t = c + \alpha y_{t-1} + \varepsilon_t$. The test statistic we use as a feature is the usual "Z-alpha" statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.pp()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

The KPSS test is based on the regression $y_t = c + \delta t + \alpha y_{t-1} + \varepsilon_t$. The test statistic we use as a feature is the usual KPSS statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.kpss()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

Autocorrelation coefficient based features

We calculate the first-order autocorrelation coefficient and the sum of squares of the first five autocorrelation coefficients of the original series, the first-differenced series, the second-differenced series, and the seasonally differenced series (for seasonal data).

A linear trend model is fitted to the time series, and the first-order autocorrelation coefficient of the residual series is calculated.

We calculate the sum of squares of the first five partial autocorrelation coefficients of the original series, the first-differenced series and the second-differenced series.

Appendix B: Distribution of MASE values shown in Table 3

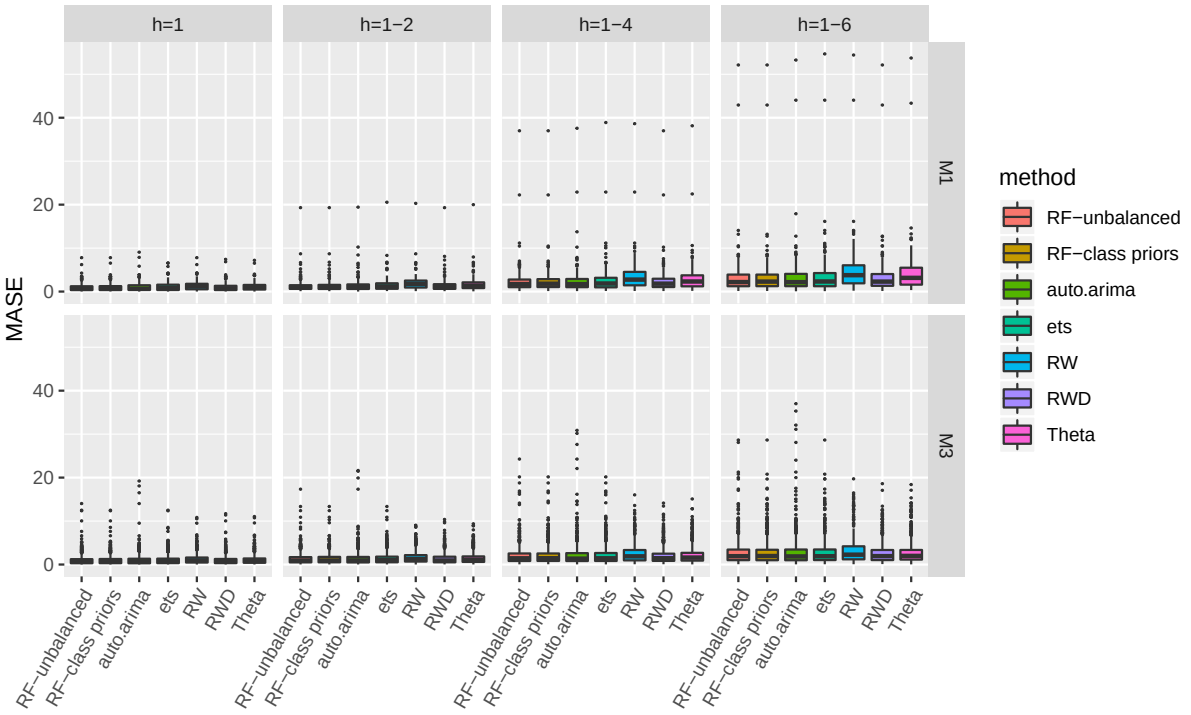

Figure 8: *Distribution of MASE values calculated over the new series for yearly data*

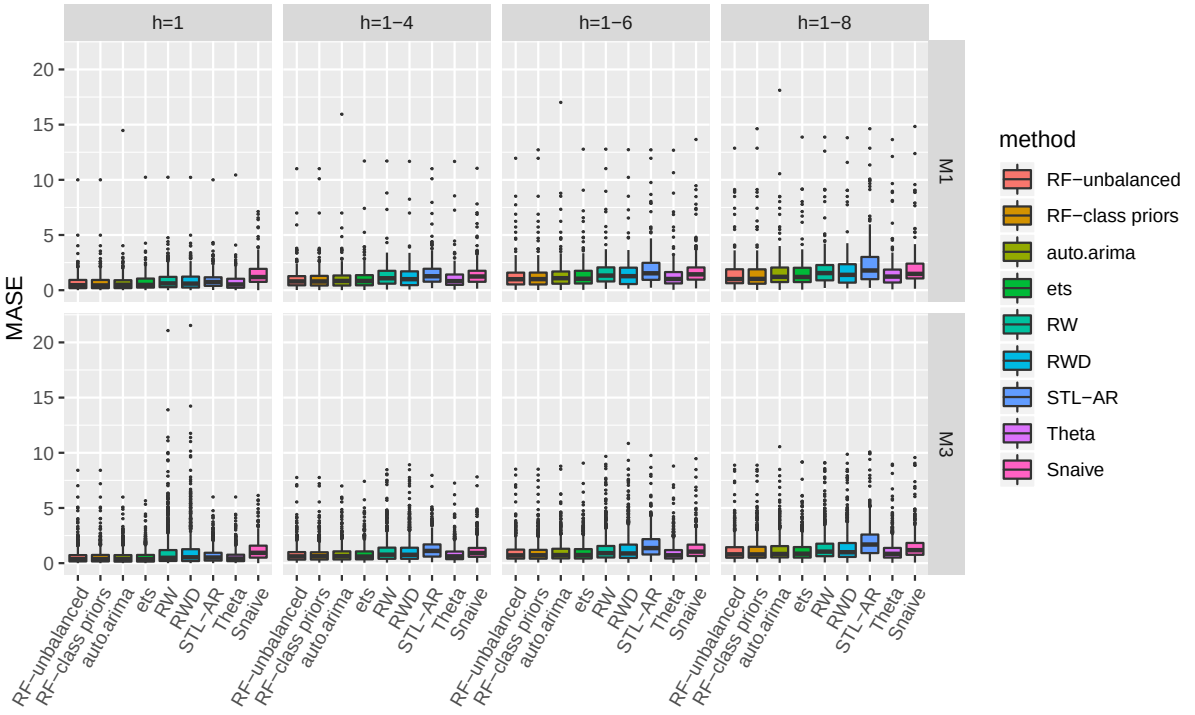

Figure 9: *Distribution of MASE values calculated over the new series for quarterly data*

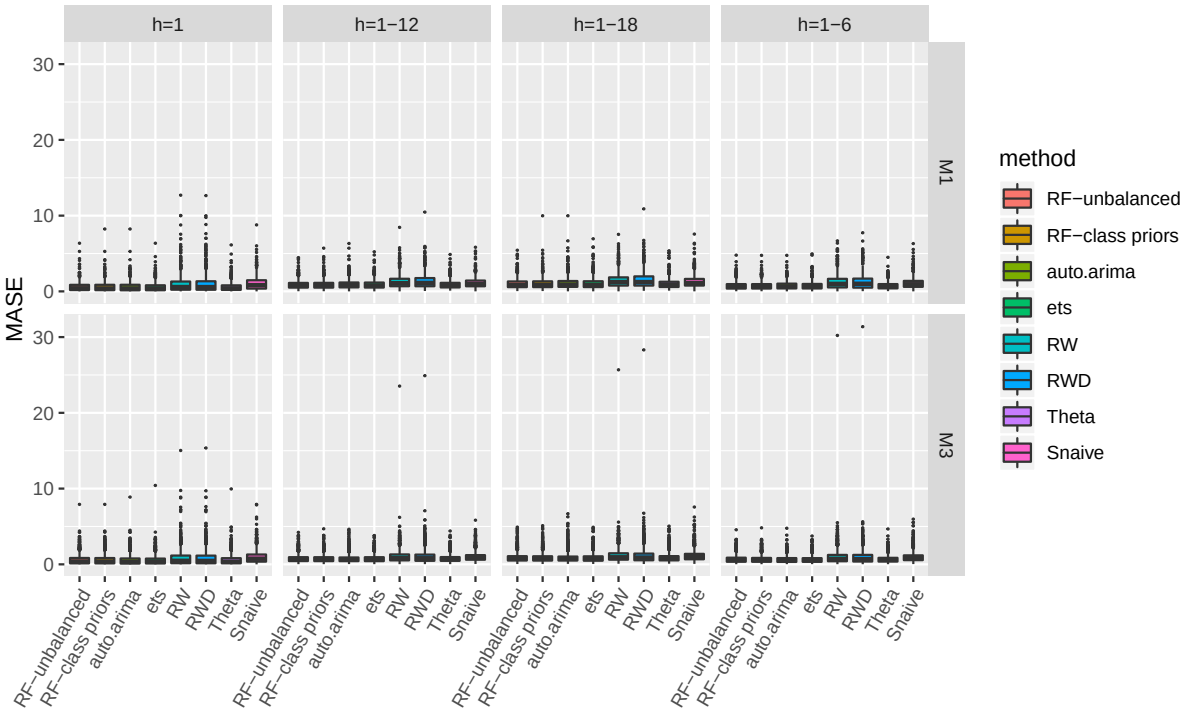

Figure 10: *Distribution of MASE values calculated over the new series for monthly data*

References

Adya, M, F Collopy, JS Armstrong & M Kennedy (2001). Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting* **17**(2), 143–157.

Armstrong, JS (2001). Should we redesign forecasting competitions? *International Journal of Forecasting* **17**(1), 542–543.

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

Breiman, L & A Cutler (2004). *Random Forests*. Version 5.1. https://www.stat.berkeley.edu/~breiman/RandomForests/.

Breiman, L, A Cutler, A Liaw & M Wiener (2018). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-14. https://cran.r-project.org/web/packages/randomForest/.

Chen, C, A Liaw & L Breiman (2004). *Using random forest to learn imbalanced data*. Tech. rep. University of California, Berkeley. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

Cleveland, RB, WS Cleveland & I Terpenning (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* **6**(1), 3.

Collopy, F & JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Fraley, C (2012). *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.4-2. https://CRAN.R-project.org/package=fracdiff.

Friedman, JH (1984). *A variable span scatterplot smoother*. Technical Report 5. Laboratory for Computational Statistics, Stanford University.

Friedman, J, T Hastie & R Tibshirani (2009). *The elements of statistical learning*. 2nd ed. New York, USA: Springer.

Fulcher, BD & NS Jones (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Goerg, GM (2016). *ForeCA: An R package for forecastable component analysis*. R package version 0.2.4. https://CRAN.R-project.org/package=ForeCA.

Goerg, G (2013). Forecastable component analysis. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp.64–72.

Guerrero, VM (1993). Time-series analysis supported by power transformations. *Journal of Forecasting* **12**, 37–48.

Haslett, J & AE Raftery (1989). Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics* **38**(1), 1–50.

Hyndman, RJ (2001). It's time to move from what to why. *International Journal of Forecasting* **17**(1), 567–570.

Hyndman, RJ (2018). *Mcomp: Data from the M-Competitions*. R package version 2.7. https://CRAN.R-project.org/package=Mcomp.

Hyndman, RJ & G Athanasopoulos (2018). *Forecasting: principles and practice*. 2nd ed. Melbourne, Australia: OTexts. https://OTexts.org/fpp2/.

Hyndman, RJ & Y Khandakar (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* **26**(3), 1–22. http://www.jstatsoft.org/article/view/v027i03.

Hyndman, RJ & AB Koehler (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting* **22**(4), 679–688.

Hyndman, RJ, AB Koehler, JK Ord & RD Snyder (2008). *Forecasting with exponential smoothing: the state space approach*. Berlin: Springer.

Hyndman, RJ, AB Koehler, RD Snyder & S Grose (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* **18**(3), 439–454.

Hyndman, RJ, E Wang & N Laptev (2015). Large-scale unusual time series detection. In: *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, pp.1616–1619.

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

Kalousis, A & T Theoharis (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis* **3**(5), 319–337.

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

Kück, M, SF Crone & M Freitag (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, pp.1499–1506.

Lawrence, M (2001). Why another study? *International Journal of Forecasting* **17**(1), 574–575.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Liaw, A & M Wiener (2002). Classification and regression by randomForest. *R News* **2**(3), 18–22. http://CRAN.R-project.org/doc/Rnews/.

Makridakis, S, A Andersen, R Carbone, R Fildes, M Hibon, R Lewandowski, J Newton, E Parzen & R Winkler (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of forecasting* **1**(2), 111–153.

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

Nuttall, AH & GC Carter (1982). Spectral estimation using combined time and lag weighting. *Proceedings of the IEEE* **70**(9), 1115–1125.

Pfaff, B (2008). *Analysis of integrated and cointegrated time series with R*. Springer.

Pfaff, B, E Zivot & M Stigler (2016). *urca: Unit root and cointegration tests for time series data*. R package version 1.3-0. https://CRAN.R-project.org/package=urca.

Prudêncio, RB & TB Ludermir (2004). Meta-learning approaches to selecting time series models. *Neurocomputing* **61**, 121–137.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Reid, DJ (1972). "A comparison of forecasting techniques on economic time series". In: *Forecasting in Action*. Birmingham, UK: Operational Research Society.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Smith-Miles, K (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)* **41**(1), 6.

Teräsvirta, T, CF Lin & CWJ Granger (1993). Power of the neural network linearity test. *Journal of Time Series Analysis* **14** (2), 209–220.

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Widodo, A & I Budi (2013). “Model selection using dimensionality reduction of time series characteristics”. Paper presented at the International Symposium on Forecasting, Seoul, South Korea. June 2013. https://goo.gl/ig2J57.

Chapter 3

Peeking inside FFORMS: Feature-based forecast model selection

Peeking inside FFORMS: Feature-based forecast model selection

Thiyanga S Talagala
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: thiyanga.talagala@monash.edu
Corresponding author

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: rob.hyndman@monash.edu

George Athanasopoulos
Department of Econometrics and Business Statistics,
Monash University, VIC 3145, Australia.
Email: george.athanasopoulos@monash.edu

7 November 2019

JEL classification: C10,C14,C22

Peeking inside FFORMS: Feature-based forecast model selection

Abstract

Features of time series are useful in identifying suitable models for forecasting. Talagala, Hyndman & Athanasopoulos (2018) proposed a classification framework, labelled FFORMS (Feature-based FORecast Model Selection), which selects forecast models based on features calculated from the time series. The FFORMS framework builds a mapping that relates the features of a time series to the "best" forecast model using a random forest. In this paper we explore what is happening under the hood of the FFORMS framework. This is accomplished using model-agnostic machine learning interpretability approaches. The analysis provides a valuable insight into how different features and their interactions affect the choice of forecast model.

Keywords: forecasting, time series, machine learning interpretability, black box models, LIME

1 Introduction

The field of time series forecasting has been evolving for several decades and has introduced a wide variety of models for forecasting. However, for a given time series the selection of an appropriate forecast model from many possibilities is not straightforward. This selection presents a challenge because each method performs best for some but not all tasks. The features of a time series are considered an important factor in identifying suitable forecasting models (Collopy & Armstrong 1992; Meade 2000; Makridakis & Hibon 2000; Wang, Smith-Miles & Hyndman 2009). However, a comprehensive description of the relationship between the features and the performance of algorithms is rarely discussed.

There have been several recent studies on the use of meta-learning approaches to automate forecast model selection based on features computed from the time series (Shah 1997; Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016). A meta-learning approach provides systematic guidance on model selection based on knowledge acquired from historical data sets. The key idea is that forecast model selection is posed as a supervised learning task.

Each time series in the metadata set is represented as a vector of features and labelled according to the best forecast model (for example, the model with the lowest mean absolute scaled error (MASE) over a test set). Then a meta-learner is trained to identify a suitable forecast model (usually a machine learning algorithm is used). In the era of big data, such an automated model selection process is necessary because the cost of invoking all possible forecast models is prohibitive. However, the existing literature suffers from limited answers to questions such as: How are features related to the property being modelled? How do features interact with each other to identify a suitable forecast model? Which features contribute most to the classification process? Addressing such questions can enhance the understanding of the relations between features and model selection outcomes. To the best of our knowledge, a very limited effort has been made to understand how the meta-learners are making their decisions and what is really happening inside these complex model structures. Providing transparency will result, building trust in the prediction results of the meta-learner.

Further, besides the goal of developing an automated forecast model selection framework, very few researchers have made an attempt to provide a description of the relationship between the features and the choice of different forecast models (Schnaars 1984; Wang, Smith-Miles & Hyndman 2009; Lemke & Gabrys 2010; Petropoulos et al. 2014, are among some exceptions). These studies are limited by the scale of problem instances used, the diversity of forecast models implemented, and the limited number of features considered to identify the relationship between features and forecast model performance.

To fill these gaps, this paper makes a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques. This paper builds on the method from our previous work Talagala, Hyndman & Athanasopoulos (2018), in which we introduced the FFORMS (Feature-based FORecast Model Selection) framework. A random forest is used to model the relationship between features and best performing forecast model. A large collection of time series is used to train the meta-learner.

In this article, we make the following contributions:

1. We extend the FFORMS framework to handle weekly, daily and hourly series. We also extend the diversity of forecast models used as class labels. The contribution of this paper differs from previously published work related to meta-learning (Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016) in three ways: i) a more extensive collection of features is used (features to handle multiple seasonality in hourly and daily

series), ii) there is a greater diversity of forecast models considered class labels, and iii) there is greater capability of handling high frequency data.

2. We analyse the application of the FFORMS framework to the M4 competition data. We generated point forecasts and prediction intervals for the M4 competition time series data, which is shown to yield accurate forecasts comparable to several benchmarks and other commonly used automated approaches of time series forecasting. Our approach achieved a high accuracy rate based on the individual forecast model selection rule.
3. The main contribution of the paper is the exploration of what is happening under the hood of the FFORMS framework, leading to an understanding of the relationship between features of time series and the choice of forecast model selection using the FFORMS framework. We call it 'Peeking inside FFORMS'. We try to answer the following questions:
   i) **Which** features are the most important?
   ii) **Where** (overall classification or only within specific classes) are they important?
   iii) **How** are they important?
   iv) **When** and **how** are features linked to the prediction outcome?
   v) **When** and **how strongly** do features interact with other features?

The remainder of the paper is structured as follows. The Section 2 outlines the extended FFORMS framework. Section 3 through Section 5 explain the three main components of the extended FFORMS framework. Results are discussed in Section 6 and Section 7. Section 8 concludes.

2 Methodological framework

Figure 1 shows the extended FFORMS framework with all additional components highlighted in yellow. There are three main components of this extended FFORMS framework. They are:

1. offline phase (blue): developing a meta-learner
2. online phase (red): using the pre-trained meta-learner to identify the best forecast model.
3. peeking inside FFORMS (yellow): gaining insights into what is happening under the hood of the FFORMS framework.

3 Offline phase

We now explain the application of the FFORMS framework to the M4 competition data. We analyse yearly, quarterly, monthly, weekly, daily and hourly series separately because of their

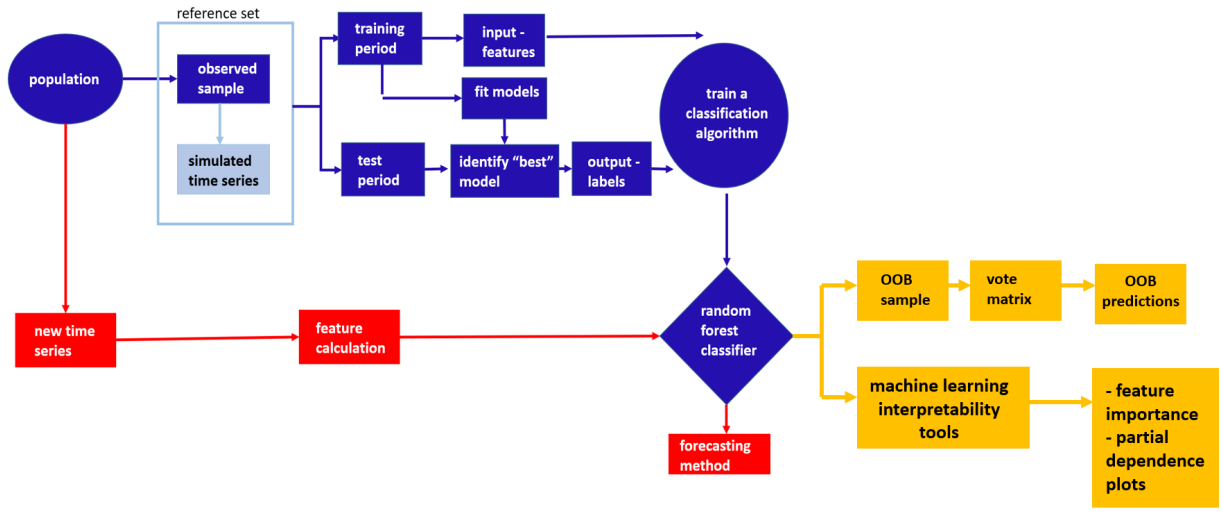

Figure 1: *Extended FFORMS (Feature-based FORecast-Model Selection) framework. The offline phase is shown in blue, the online phase in red and the peeking inside FFORMS is shown in yellow.*

differences in frequencies and hence the appropriateness of different features and forecast models we considered as class labels.

3.1 Reference set

We call the collection of time series used for training the meta-learner the 'reference set'. The reference set consists of two sets of time series: i) the observed sample and ii) the simulated time series.

3.1.1 Observed sample

We use the time series from the M1, M3 and M4 competitions as the observed sample. Note that from the M4 competition a randomly selected subset of time series is used for the observed sample. The rest is used as a validation set to evaluate the performance of the meta-learner. Note that we did not use the test period of each series of the M4 competition in our observed sample or validation set because it was not available to the competitors during the running of the M4 competition.

3.1.2 Simulated time series

As described in Talagala, Hyndman & Athanasopoulos (2018), we augment the reference set by adding multiple time series simulated using the training period of each series in the M4

competition. We use several automated algorithms to simulate multiple time series. The `ets()` and `auto.arima()` functions available in the forecast package in R (Hyndman et al. 2018) are used to simulate yearly, quarterly and monthly data from exponential smoothing and ARIMA models. The `stlf()` function, also available in the forecast package, is used to simulate multiple time series based on a multiple seasonal decomposition approach. Using the above functions, we fit models to each time series in the M4 database and then simulate multiple time series from the selected models. In this experiment the length of the simulated time series is set to be equal to the length of the training period specified in the M4 competition plus the length of the forecast horizon specified in the competition. For example, the series with id Y13190 contains a training period of length 835. The length of the simulated series generated from this series is equal to 841 (835 + 6). Before simulating time series from daily and hourly series, we convert the time series into multiple seasonal time series (msts) objects. For daily time series with a length of less than 366, the frequency is set to 7. Longer time series are converted to multiple seasonal time series objects with frequencies set to 7 and 365.25. For hourly series, we set the frequencies to 24 and 168 to handle multiple frequencies corresponding to time-of-day pattern and time-of-week pattern respectively. We should re-emphasise that all the observed time series and the simulated time series form the reference set used to build our meta-learner. Once we have created the reference set for random forest training, we split each time series in the reference set into training period and test period.

3.2 Input: features

The FFORMS framework operates on features calculated from the time series. For each time series in the reference set, features are calculated based on the training period of the time series. The description of the features calculated under each frequency category is shown in Table 1. A comprehensive description of the features used in the experiment is given in Talagala, Hyndman & Athanasopoulos (2018).

3.3 Output: class-labels

The description of class labels considered under each frequency is shown in Table 2. Note that these added to Talagala, Hyndman & Athanasopoulos (2018). Most of the labels given in Table 2 are self-explanatory. In stlar, mstlets and mstlarima, first an STL decomposition or a multiple seasonal decomposition (for mstlarima and mstlets) is applied to the time series and then seasonal naive is used to forecast the seasonal component. Then AR, ETS and ARIMA models are used to forecast the seasonally adjusted data. We fit the corresponding models outlined in Table 2 to each series in the reference set. The models are estimated using the

Table 1: *Time series features*

	Feature	Description	Y	Q/M	W	D/H
1	T	length of time series	✓	✓	✓	✓
2	trend	strength of trend	✓	✓	✓	✓
3	seasonality_q	strength of quarterly seasonality	-	✓	-	-
4	seasonality_m	strength of monthly seasonality	-	✓	-	-
5	seasonality_w	strength of weekly seasonality	-	-	✓	✓
6	seasonality_d	strength of daily seasonality	-	-	-	✓
7	seasonality_y	strength of yearly seasonality	-	-	-	✓
8	linearity	linearity	✓	✓	✓	✓
9	curvature	curvature	✓	✓	✓	✓
10	spikiness	spikiness	✓	✓	✓	✓
11	e_acf1	first ACF value of remainder series	✓	✓	✓	✓
12	stability	stability	✓	✓	✓	✓
13	lumpiness	lumpiness	✓	✓	✓	✓
14	entropy	spectral entropy	✓	✓	✓	✓
15	hurst	Hurst exponent	✓	✓	✓	✓
16	nonlinearity	nonlinearity	✓	✓	✓	✓
17	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓	✓	-
18	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓	✓	-
19	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓	-	-
20	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓	-	-
21	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓	-	-
22	ur_pp	test statistic based on Phillips-Perron test	✓	-	-	-
23	ur_kpss	test statistic based on KPSS test	✓	-	-	-
24	y_acf1	first ACF value of the original series	✓	✓	✓	✓
25	diff1y_acf1	first ACF value of the differenced series	✓	✓	✓	✓
26	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓	✓	✓
27	y_acf5	sum of squares of first 5 ACF values of original series	✓	✓	✓	✓
28	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓	✓	✓
29	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓	✓	✓
30	sediff_acf1	ACF value at the first lag of seasonally-differenced series	-	✓	✓	✓
31	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓	✓	✓
32	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓	✓	✓
33	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	✓	✓	✓
34	lmres_acf1	first ACF value of residual series of linear trend model	✓	-	-	-
35	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓	✓	✓
36	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓	✓	✓
37	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓	✓	✓

training period for each series, and forecasts are produced for the test periods based on "a fixed origin" evaluation.

According to the *M4 Competitor's Guide* (2018), in the M4 competition the forecast accuracy is evaluated based on the MASE and the symmetric mean absolute percentage error (sMAPE). Hence, to identify the best forecast model for each time series in the reference set, we combine MASE and the sMAPE calculated over the test set. More specifically, for each series, both forecast error measures, MASE and sMAPE, are calculated for each of the forecast models. Each of these is standardised by the median MASE and median sMAPE, calculated across the forecast models. The model with the lowest average value of the scaled MASE and scaled sMAPE is selected as the output class label. The last step of the offline phase of the framework is to train a

Table 2: *Class labels*

Class label	Description	Y	Q/M	W	D/H
wn	white noise process	✓	✓	✓	✓
ARMA	AR, MA, ARMA processes	✓	✓	✓	-
ARIMA	ARIMA process	✓	✓	✓	-
SARIMA	seasonal ARIMA	-	✓	✓	-
rwd	random walk with drift	✓	✓	✓	✓
rw	random walk	✓	✓	✓	✓
theta	standard theta method	✓	✓	✓	✓
stlar		-	✓	✓	✓
ETS_NTNS	ETS without trend and seasonal components	✓	✓	-	-
ETS_T	ETS with trend component and without seasonal component	✓	✓	-	-
ETS_DT	ETS with damped trend component and without seasonal component	✓	✓	-	-
ETS_TS	ETS with trend and seasonal component	-	✓	-	-
ETS_DTS	ETS with damped trend and seasonal component	-	✓	-	-
ETS_S	ETS with seasonal component and without trend component	-	✓	-	-
snaive	seasonal naive method	-	✓	✓	✓
tbats	TBATS forecasting	-	✓	✓	✓
nn	neural network time series forecasts	✓	✓	✓	✓
mstlets		-	-	✓	✓
mstlarima		-	-	-	✓

meta-learner. A random forest algorithm is used to train the meta-learner.

4 Online phase

The online phase of the algorithm involves generating point forecasts and 95% prediction intervals for the new series or the future values observed time series. We calculate point forecasts and prediction intervals over the test period of M4 competition data. First, the corresponding features are calculated based on the full length of the training period provided by the M4 competition. Second, point forecasts and 95% prediction intervals are calculated based on the predicted class labels. We should note that all negative forecasts are set to zero. According to the *M4 Competitor's Guide* (2018), the performance of prediction intervals are evaluated based on the mean scaled interval score (MSIS).

5 Peeking inside FFORMS

The main objective of this paper is to explore the nature of the relationship between features and forecast model selection learned by the FFORMS framework. More specifically, it is to identify which of the features are important for model predictions and how different features and their interactions led to the different choices. First, we explore the global role of features in the choice of different forecast model selection. Global interpretability evaluates the behaviour of a model on an entire data set. The global perspective of model interpretation helps users to understand the overall modelled relationship between features and the FFORMS outcome. In the following

subsections, we provide a description of tools we use to explore the global perspective of the FFORMS meta-learners.

5.1 Visualise patterns learned by the meta-learner

A useful by-product of random forest model is out-of-bag (OOB) observations. OOB observations are the observations not included in building a given tree. In general, each tree uses only around one third of observations in its construction. Further, each tree is grown based on different bootstrap samples; hence, each tree has a different set of OOB observations. We use a vote matrix calculated based on OOBs to visualise patterns learned by the random forest. The vote matrix ($N \times P$; N is the total number of observations; P is the number of classes) contains the proportion of times each observation was classified to each class based on OOB observations.

5.2 Feature importance

Let $\mathcal{P} = \{(\mathbf{f_i}, z_i)\}_{i=1}^{N}$ be the historical data set we use to train a classifier. Consider a d-dimensional feature vector $\mathbf{F} = (F_1, F_2, ..., F_d)$ and a dependent variable, the best forecasting method for each series Z. Let $\mathcal{G}$ be the unknown relationship between $\mathbf{F}$ and Z. Zhao & Hastie (2017) call this 'law of nature'. Inside the FFORMS framework, the random forest algorithm tries to learn this relationship using the historical data we provided. We denote the predicted function as g.

Jiang & Owen (2002) explain variable importance under three different views: i) causality (change in the value of Z for an increase or decrease in the value of F_i), ii) contribution of F_i based on out-of-sample prediction accuracy and iii) face value of F_i on prediction function g. For example, in linear regression models estimated coefficients of each predictor can be considered a measure of variable importance. See Jiang & Owen (2002) for comparable face value interpretation for machine learning models. In this paper we use the first two notions of variable importance. Partial dependency functions and individual conditional expectation curves are used to explore the 'causality' notion of variable importance, while mean decrease in Gini coefficient and permutation-based variable importance are used to capture the second notion of variable importance (feature contribution to the predictive accuracy) (Zhao & Hastie (2017)). We introduce each of these variable importance measures below.

5.2.1 Mean decrease in Gini coefficient

Mean decrease in Gini coefficient is a measure of how each feature contributes to the homogeneity of the nodes and leaves in the resulting random forest proposed by Breiman (2001).

5.2.2 Permutation-based variable importance measure

The permutation-based variable importance introduced by Breiman (2001) measures the prediction strength of each feature. This measure is calculated based on the out-of-bag (OOB) observations. The calculation of variable importance is formalised as follows: Let $\bar{\mathcal{B}}^{(k)}$ be the OOB sample for a tree k, with $k \in \{1, ..., ntree\}$, where $ntree$ is the number of trees in the random forest. Then the variable importance of variable F_j in k^{th} tree is:

$$VI^{(k)}(F_j) = \frac{\sum_{i \in \bar{\mathcal{B}}^{(k)}} I(\gamma_i = \gamma^k_{i,\pi_j})}{|\bar{\mathcal{B}}^{(k)}|} - \frac{\sum_{i \in \bar{\mathcal{B}}^{(k)}} I(\gamma_i = \gamma^k_i)}{|\bar{\mathcal{B}}^{(k)}|},$$

where γ^k_i denotes the predicted class for the i^{th} observation before permuting the values of F_j and γ^k_{i,π_j} is the predicted class for the i^{th} observation after permuting the values of F_j. The overall variable importance score is calculated as:

$$VI(F_j) = \frac{\sum_{k=1}^{ntree} VI^{(k)}(F_j)}{ntree}.$$

Permutation-based variable importance measures provide a useful starting point for identifying the relative influence of features on the predicted outcome. However, they provide little indication of the nature of the relationship between the features and model outcome. To gain further insights into the role of features inside the FFORMS framework we use the partial dependence plot (PDP) introduced by Friedman, Popescu, et al. (2008).

5.2.3 Partial dependence plot (PDP) and variable importance measure based on PDP

Partial dependence plots can be used to graphically examine how each feature is related to the model prediction while accounting for the average effect of other features in the model. Let F_s be the subset of features we want to examine for partial dependencies and F_c be the remaining set of features in F. Then g_s, the partial dependence function on F_s is defined as

$$g_s(F_s) = E_{f_c}[g(f_s, F_c)] = \int g(f_s, f_c) dP(f_c).$$

In practice, PDP can be estimated from a training data set as

$$\bar{g_s}(f_s) = \frac{1}{n} \sum_{i=1}^{n} g(f_s, F_{iC}),$$

where n is the number of observations in the training data set. A partial dependency curve can be created by plotting the pairs of $\{(f_s^k, \bar{g}_s(f_{sk}))\}_{k=1}^m$ defined on a grid of points $\{f_{s1}, f_{s2}, \ldots, f_{sm}\}$ based on F_s. The FFORMS framework has treated the forecast model selection problem as a classification problem. Hence, in this paper partial dependency functions display the probability of a certain class occurring given different values of the feature F_s.

Greenwell, Boehmke & McCarthy (2018) introduce a variable importance measure based on the partial dependency curves. The idea is to measure the 'flatness' of partial dependence curves for each feature. A feature with a PDP curve that is flat, relative to the other features, does not have much influence on the predicted value. The flatness of the curve is measured using the standard deviation of the values $\{\bar{g}_s(f_{sk})\}_{k=1}^m$.

5.2.4 Individual conditional expectation (ICE) curves and variable importance measure based on ICE curves

While partial dependency curves are useful in understanding the estimated relationship between features and the predicted outcome in the presence of substantial interaction between features, they can be misleading. Goldstein et al. (2015) propose individual conditional expectation (ICE) curves to address this issue. Instead of averaging $g(f_s, F_{iC})$ over all observations in the training data, an ICE curve generates the individual response curves by plotting the pairs $\{(f_s^k, g(f_{sk}, F_{iC}))\}_{k=1}^m$ defined on grid of points $\{f_{s1}, f_{s2}, \ldots, f_{sm}\}$ based on F_s. In other words, the partial dependency curve is simply the average of all the ICE curves.

This method is similar to the PDP-based variable importance scores above, but is based on measuring the 'flatness' of the individual conditional expectation curves. We calculate standard deviations of each ICE curve. We then compute an ICE-based variable importance score, which is simply the average of all the standard deviations. A higher value indicates a higher degree of interactivity with other features.

5.2.5 Ranking of features based on feature importance measures

To identify important class-specific features we rank features in three different ways: i) based on permutation-based variable importance, ii) based on partial dependence functions and iii) based on ICE curves. We consider 25 features for yearly data. The feature that shows the highest importance is ranked 25, the second highest is ranked 24, and so on. Finally, for each feature a mean rank is calculated based on the rankings of the three measures. Similarly, the overall feature importance is evaluated based on the permutation-based variable importance measure and the Gini coefficient-based feature importance measure.

5.3 Relationship between most important features and the choice of forecast model selection

The partial-dependence curves, along with their confidence intervals, are used to visualise the relationship between top features and the choice of forecast model selection.

5.4 Assessment of interaction effect

Friedman's H-statistic (Friedman, Popescu, et al. 2008) is used to test the presence of interaction between all possible pairs of features. This statistic is computed based on the partial dependence functions. For two-way interaction between two specific variables Y_j and Y_k, Friedman's H-statistic is defined as follows:

$$H_{jk}^2 = \sum_{i=1}^{n}[\bar{g}_s(f_{ij}, f_{jk}) - \bar{g}_s(f_{ij}) - \bar{g}_s(f_{ik})]^2 / \sum_{i=1}^{n}\bar{g}_s^2(f_{ij}, f_{jk}).$$

Friedman's H-statistic measures the fraction of variance of two-variable partial dependency, $\bar{g}_s(f_{ij}, f_{jk})$ not captured by the sum of the respective individual partial dependencies, $\bar{g}_s(f_{ij}) + \bar{g}_s(f_{ik})$. In addition to Friedman's H-statistic, we use the PDP of two variables to visualise the interaction effects. Similarly, Friedman's H-statistic for testing whether a specific feature interacts with any other feature can be estimated by

$$H_j^2 = \sum_{i=1}^{n}[g(\boldsymbol{f_i}) - \bar{g}_s(f_{ij}) - \bar{g}_s(f_{i/j})]^2 / \sum_{i=1}^{n} g^2(\boldsymbol{f_i}).$$

Note that the, PD plots, ICE curves and PD- and ICE-associated measures and Friedman's H-statistic are computationally intensive to compute, especially when there are a large number of observations in the training set. Hence, in our experiments ICE- and PDP-based variable importance measures are computed based on the subset of randomly selected training examples.

5.5 Local interpretable model-agnostic explanations (LIME)

Global interpretations help us to understand the entire modelled relationship. Local interpretations help us to understand the predictions of the model for a single instance or a group of similar instances. In other words, this allows users to zoom into a particular instance or a subset and explore how different features affect the resulting prediction. We use the Local Interpretable Model-agnostic Explanations (LIME) approach introduced by Ribeiro, Singh & Guestrin (2016) for explaining individual predictions. This approach relies on the assumption that 'every complex model is linear on a local scale'. It locally approximates the complex black box model with a simple interpretable model. Ribeiro, Singh & Guestrin (2016) highlighted

that globally important features may not be important in the local context, and vice versa. The algorithm steps can be summarized as follow:

1. Select an observation of interest for which we need black box prediction explanations.
2. Create a permuted data set based on the selected observation. A permuted data set is created by making slight modifications to the features of selected observations.
3. Obtain similarity scores by calculating distance between permuted data and selected observation.
4. Obtain predicted outcomes for all permuted data using the black box model.
5. Select q number of features best describing the black box model outcome. This can be accomplished by applying feature selection algorithms such as ridge regression and lasso regression.
6. Fit a simple linear model to the permuted data based on q selected features, similarity scores in step 3 as weights and complex model prediction outcomes in step 4 as response variable.
7. Use the estimated coefficients of the simple linear model to explain how the local behaviour corresponds to the selected observation in step 1.

An alternative for explaining local behaviour of complex models is proposed by Lundberg & Lee (2017). The approach is developed based on game theory, it is named 'Shapley values'.

Table 3: *The performance of FFORMS on the M4 competition data based on point forecasts (based on MASE) and prediction intervals (based on MSIS)*

Point Forecasts (Mean Absolute Scaled Error (MASE))						
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.17	1.20	0.98	2.31	3.57	0.84
auto.arima	3.40	1.17	0.93	2.55	-	-
ets	3.44	1.16	0.95	-	-	-
theta	3.37	1.24	0.97	2.64	3.33	1.59
rwd	3.07	1.33	1.18	2.68	3.25	11.45
rw	3.97	1.48	1.21	2.78	3.27	11.60
nn	4.06	1.55	1.14	4.04	3.90	1.09
stlar	-	2.02	1.33	3.15	4.49	1.49
snaive	-	1.66	1.26	2.78	24.46	2.86
tbats	-	1.19	1.05	2.49	3.27	1.30
wn	13.42	6.50	4.11	49.91	38.07	11.68
mstlarima	-	-	-	-	3.84	1.12
mstlets	-	-	-	-	3.73	1.23
combination (mean)	4.09	1.58	1.16	6.96	7.94	3.93
M4-1st	**2.98**	1.12	**0.88**	2.36	3.45	0.89
M4-2nd	3.06	**1.11**	0.89	**2.11**	3.34	**0.81**
M4-3rd	3.13	1.12	0.91	2.16	**2.64**	0.87
Prediction Intervals (Mean Scaled Interval Score (MSIS))						
FFORMS	39.79	11.24	9.82	**20.84**	36.36	8.0
M4-1st	**23.89**	**8.55**	**7.20**	22.03	**26.28**	**7.92**
M4-2nd	27.47	9.38	8.65	21.53	34.37	18.50
M4-3rd	not submitted					
naive	56.55	14.07	12.30	26.35	32.55	71.24

Note: Bold signifies the best performing method.

6 Evaluation based on the M4 competition data

Table 3 shows the performance of the FFORMS approach on the M4 competition data. The point forecasts and prediction intervals are evaluated based on the test period of each series. The point forecasts are evaluated based on the MASE computed for each forecast horizon, and then by averaging the MASE errors across all series corresponding to each frequency category. Similarly, MSIS is used to evaluate the prediction intervals. The results are compared against several benchmarks and the top three places of the M4 competition. The top-ranking methods of the M4 competition are based on some kind of combination approach. The first ranking method is based on a hybrid approach that produces forecasts based on both ETS and machine learning approaches. The second and third places are based on a combination of nine and seven different forecast models. The second and third approaches are time-consuming because forecasts from all candidate models must be computed. The results of the FFORMS approach are based on individual forecasts. According to the Table 3, we can see the FFORMS approach achieved comparable performance in a much more cost and time-effective way. Based on the FFORMS approach, we provided an entry to the M4 competition. After submission we found a bug in our implementation of hourly data, hence only the hourly data results are different from the published results of the M4 competition. The yearly, quarterly, monthly, weekly and daily result are same as the published results. This completes the evaluation of the FFORMS framework. The main question we have now is: 'Can we trust the FFORMS framework if we do not know how it works?' In the next section we present results showing what is happening inside the FFORMS framework.

7 What is happening under the hood of FFORMS?

7.1 Yearly series

The vote-matrix information for the yearly data is presented in Figure 2. Figure 2 indicates three primary patterns: i) the distributions of correctly classified classes dominate, indicating a good classification of the meta-learner, and the associated outliers of the boxplots indicate some series are correctly classified with high probability; ii) irrespective of the class labels, the random walk with drift has a high chance of being selected for all series (the results of Section 6 also show random walk with drift giving very high prediction accuracy with yearly series); and iii) the FFORMS framework successfully learned the similarities and dissimilarities between the classes itself. For example, within ETS models with a trend component (ETS_T, 3rd panel), disparities

between forecast models with trend components and forecast models without trend components (wn, ARMA, ETS_NTNS) are immediately visible.

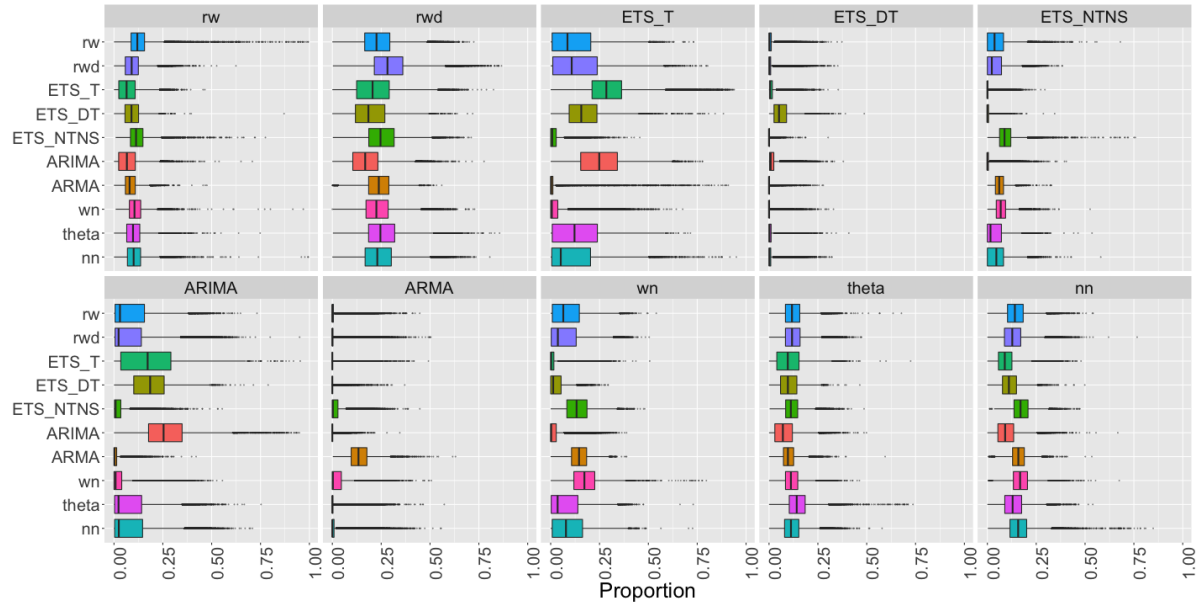

Figure 2: *Visualisation of the vote matrix based on OOB sample for yearly random forest. Each panel shows the predicted class from the random forest. The X-axis denotes the proportion of times each time series was classified to each class. The colours of boxplots correspond to class label of the best forecast model identified based on MASE and sMAPE. On each row, distribution of correctly classified class dominates, indicating a good classification of the meta-learner.*

Figure 3 shows the importance of each of the features we considered in the overall classification mechanism of the FFORMS, as well as within each class. The first panel gives the ranks of the overall variable importance of all features and the other panels show the class-specific feature rankings. The main point here is strength of trend, test statistic based on the Phillips-Perron unit root test, the first ACF value of the differenced series and linearity are the most important features across all categories. In addition, the first ACF value of residual series of linear trend model (lmres_acf1) appears to be important within all classes. In addition to linearity, the other features related to different types of trend (damped trend, measured by beta, and exponential trend, measured by curvature) are assigned a very high importance within ETS_T, ETS_DT and ETS_NTNS, which handle different versions of trend within the ETS family. Spikiness appears to be an important feature in the overall classification, and even though it does not appear to be among the top five features, spikiness has been assigned a relatively high importance within all categories. The reason for the high importance of spikiness within the overall classification is the high mean decrease in Gini coefficient. This suggests that it is the contribution of spikiness to the homogeneity of the nodes and leaves in the resulting random forest that makes it important, rather than its effect of causality. The length of time series (T) is assigned a very low importance

within all categories.

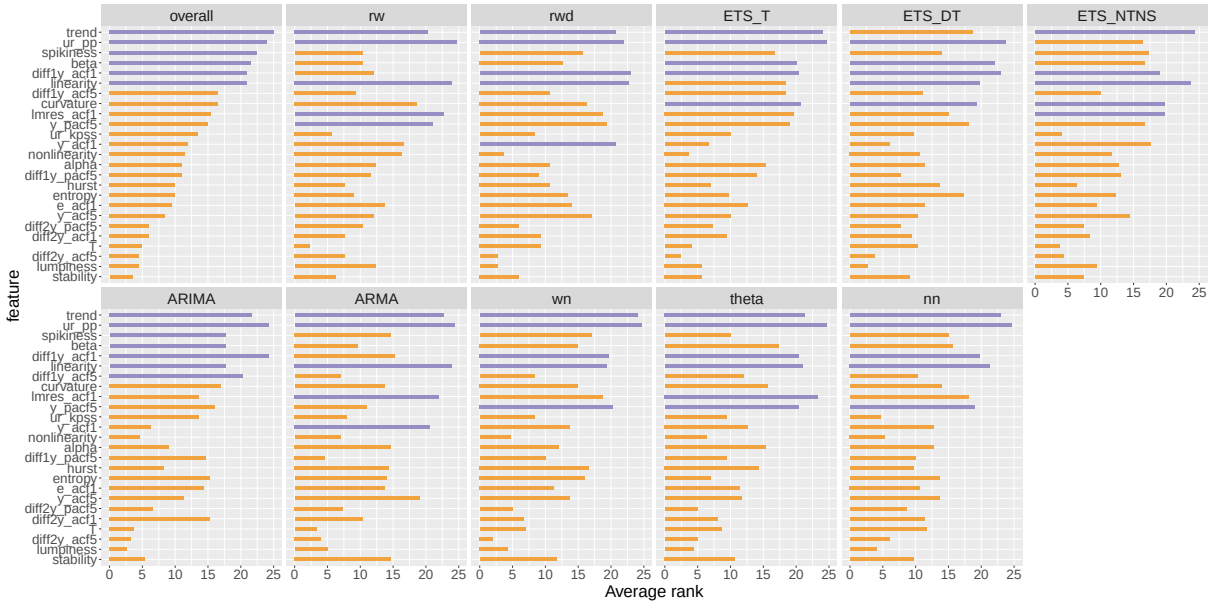

Figure 3: *Feature importance plot for yearly series. Overall feature importance (top left plot) is evaluated based on two measures: i) permutation-based variable importance measure and ii) mean decrease in Gini coefficients. Class-specific feature importance is evaluated based on three measures: i) permutation-based variable importance, ii) PD-based variable importance measure, and iii) ICE-based variable importance measure. Longer bars indicate more important features. Top 5 overall features are highlighted in purple. Strength of trend appears to be the most important feature.*

Figure 4 to Figure 6 show the partial dependency curves and associated confidence intervals of the top three features that are selected most in each class. In all three cases, within each class the displayed relationship is consistent with the theoretical expectations. For example, a high negative value of the Phillips-Perron test statistic indicates a stationarity of the series, while a positive value of the test statistic indicates nonstationarity of the series. According to Figure 4, we can see that the probability of selecting an ETS model with a trend component and ARIMA models increases steadily as ur_pp increases, while the opposite relationship could be observed for ARMA and WN. According to Figure 5, we can see that the information about the strength of trend in the series is very important when selecting a forecast model. The probability of selecting an ETS model without trend components (ETS_NTNS), WN and ARMA decreases when the strength of trend is extremely high, while the opposite relationship can be observed for other classes. According to Figure 6 random walk with drift and ARMA classes are highly sensitive to the value of linearity around 0. According to Friedman's H-statistic, diff1y_acf1 and lmres_acf1 show high levels of two-way interactivity. The associated two-way partial dependency plots are shown in Figure 7. The classes rw, rwd, ETS_NTNS, ARIMA and nn show interactivity. The plots suggest that the probability of selecting the corresponding model changes according to the changes in both diff1y_acf1 and lmres_acf1. Within the neural network class we can see that

when lmres_acf1 arrives closer to 1, lmres_acf1 does not interact with diff1y_acf1. ETS_T shows low interactivity, irrespective of the value of diff1y_acf1 series, with a low value of lmres_acf1 leading to a high chance of selecting ETS_T model. Within ETS_T the colours of the bands are constant throughout the domain. However, for rw, rwd, ARIMA, theta and neural networks the strength of colour varies according to the levels of both diff1y_acf1 and lmres_acf1 indicating an interactivity between the two features. The interactivity pattern within wn class is not visible at a 0-0.21 scale but it is visible within a 0-1 scale. The associated two-way partial dependency plot of wn series is shown in Figure 40 (see in the Appendix B). This suggests that two-way interactivity easily separates wn from the rest.

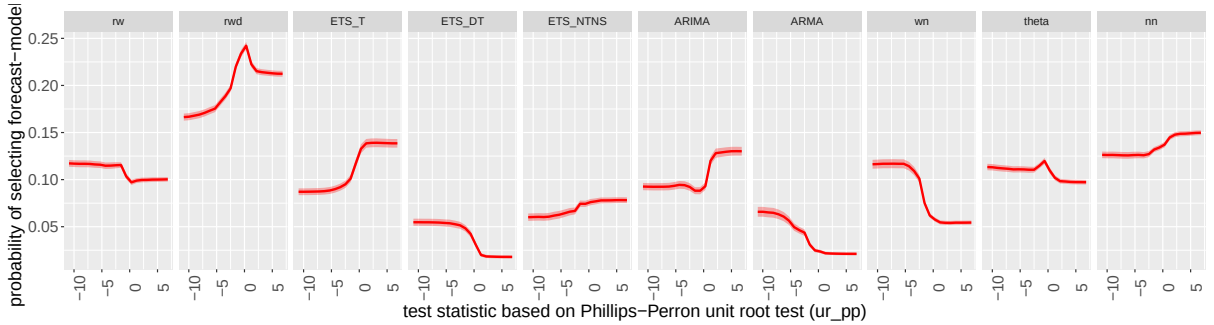

Figure 4: *Partial dependence plots for ur_pp. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. All classes show a turning point in the relationship around zero.*

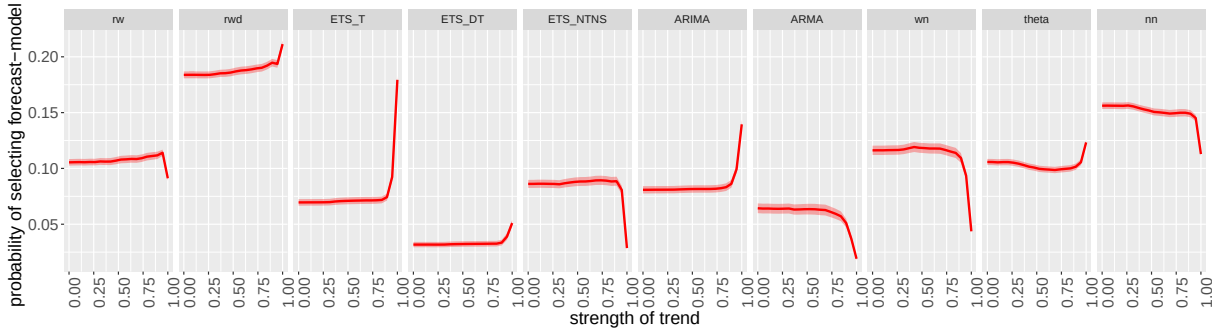

Figure 5: *Partial dependence plots for trend. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting ETS models without a trend component and stationary models (WN and ARMA) decreases for an extremely high value of trend.*

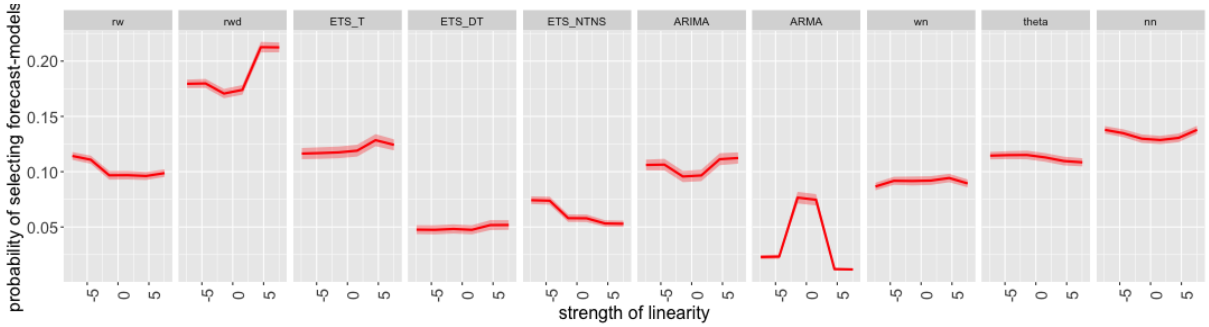

Figure 6: *Partial dependence plots for linearity. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Random walk with drift and ARMA classes are highly sensitive to the value of linearity around 0.*

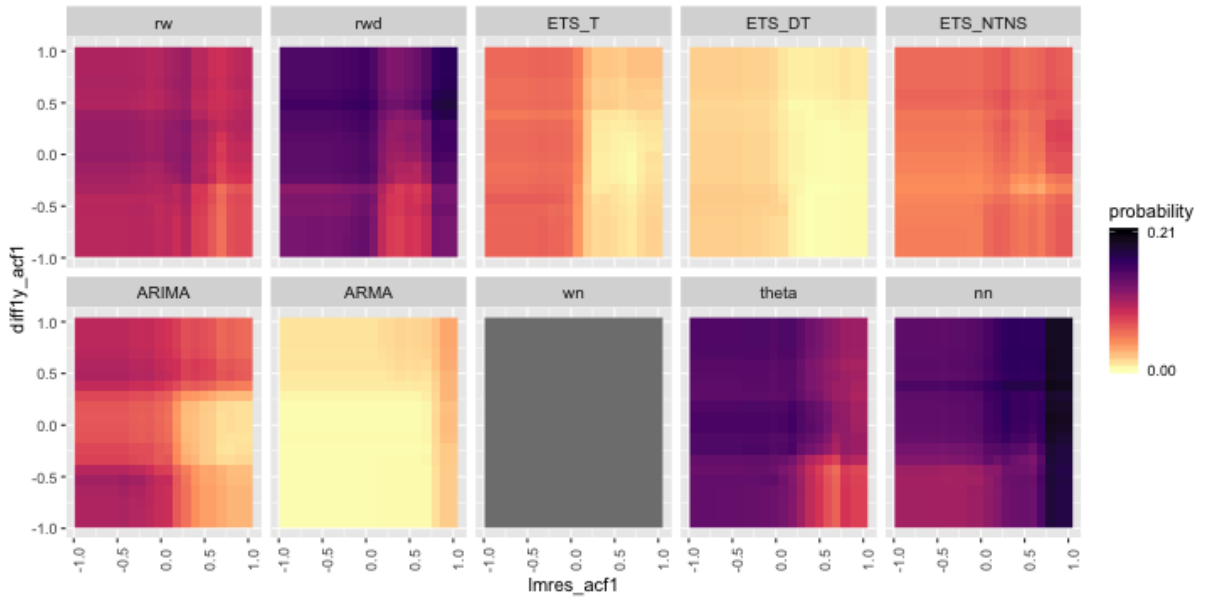

Figure 7: *Partial dependence of diff1y_acf1 and lmres_acf1 for yearly data. Dark regions show the high probability of belonging to the corresponding class shown in the plot title. Random walk with drift (rwd) and theta show a similar pattern of interactivity between diff1y_acf1 and lmres_acf1.*

7.2 Quarterly series

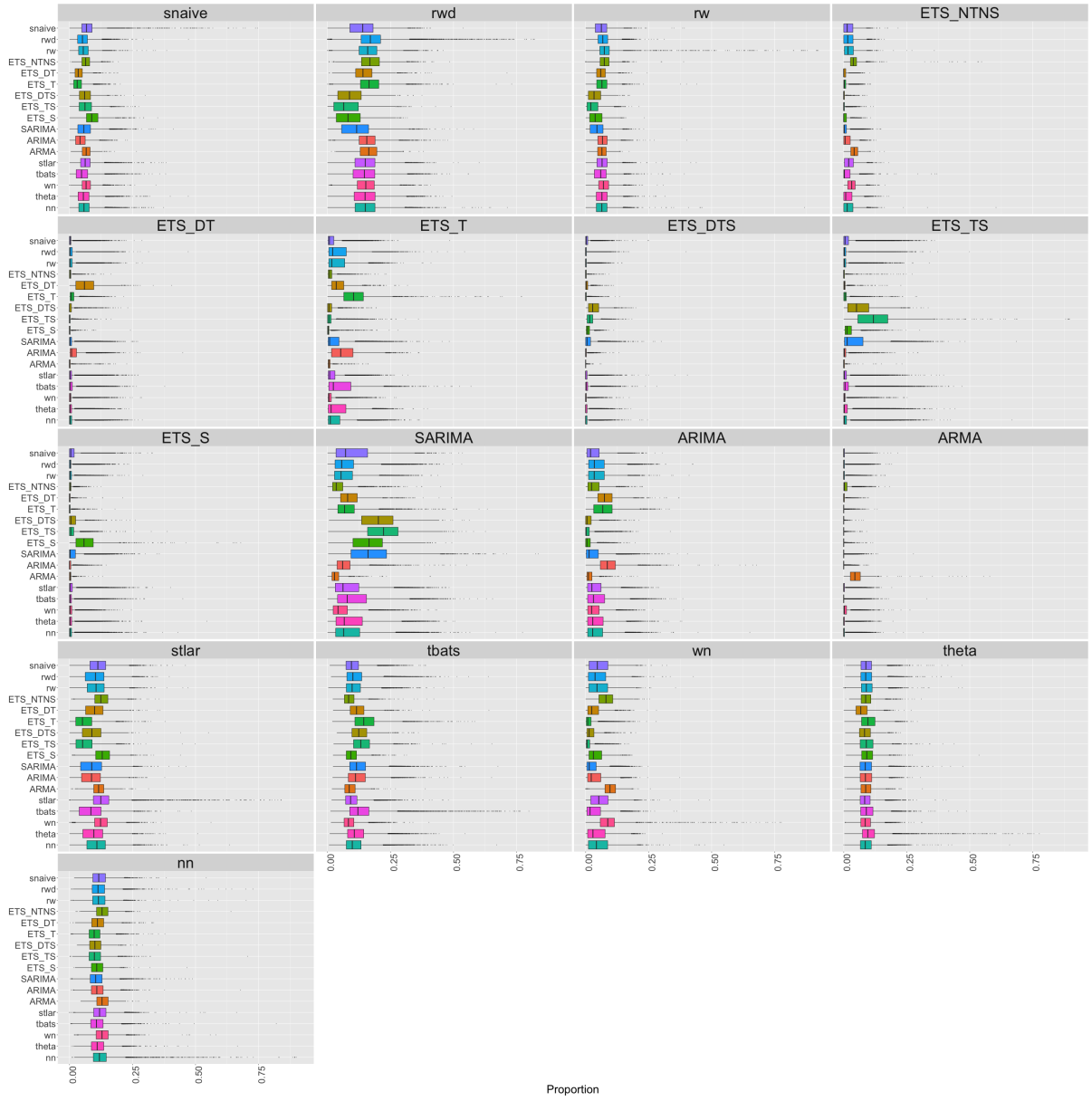

Figure 8: *Visualisation of the vote matrix based on OOB sample for quarterly random forest. Each panel shows the predicted class from the random forest. The X-axis denotes the proportion of times each time series was classified to each class. The colours of boxplots correspond to class label of the best forecast model identified based on MASE and sMAPE. The disparities between dominating class and non-dominating classes are immediately apparent within ETS model classes.*

Figure 8 shows the vote-matrix of the random forests for quarterly series based on OOB observations. The disparities between dominating class and non-dominating classes are immediately apparent within ETS_DT, ETS_T, ETS_DTS, ETS_TS, ETS_S and ARMA classes. Within each panel, the outliers associated with dominating distribution indicate some series are correctly classified with very high probability. Further, except for ETS_DTS, ETS_TS, ETS_S and SARIMA, all other series have a high chance of being classified into a random walk with drift (rwd) class.

Within the theta and nn all distributions level at similar proportionalities as they both represent a general class of forecast models. Overall, the random forest successfully learned the similarities and dissimilarities between different forecast models.

Figure 9 shows the feature importance plots for the quarterly series. The features seasonality_q, trend, diff1y_pacf5, linearity and spikiness are the most important features across all categories. Even though lumpiness does not appear as a top five feature within classes, it appears to be an important feature in the overall classification process and relatively high ranks are assigned within many classes. Except for, ETS_DT, ETS_DTS and ETS_TS, the features nonlinearity, hwalpha and hwbeta are assigned a very low importance within all other classes.

Figure 10 to Figure 12 show the partial dependency curves and associated confidence intervals of the top three features that are selected most in each class. According to Figure 10, it is immediately apparent that the probability of selecting a seasonal forecast model increases as the strength of seasonality increases. According to Figure 10 and Figure 11, probability of selecting SARIMA models drastically changes according to the change of seasonality_q, and diff1y_pacf5. The partial dependence plot of trend is very similar to the observed patterns in yearly data.

According to Friedman's H-statistic (>0.5), curvature and e_acf1 is the most commonly appearing two-way interactivity. The associated partial dependency plots are shown in Figure 13. The interactivity pattern of wn class is visible only according to the 0-1 scale, which suggests the two-way interactivity easily separates the wn class from the rest. Conversely, the interactivity pattern with ETS_NTNS, ETS_DT, ETS_T, ARMA, ARIMA is not visible within the 0-0.125 scale. The reason could be that these models do not have a parameter to handle seasonal components. Hence, the probability of selecting these models is relatively low compared with other models. Hence, the pattern of interactivity is not visible within the 0-0.125 scale. According to Figure 13, we can see that the pattern of interactivity does not completely change the individual relationship of the feature with the predicted outcome, but interactivity influences the probability of selecting the corresponding forecast model. For example, stlar models generally have a high change of being selected with series with high curvature value. However, this probability changes according to the values of e_acf1. Within ETS_S when curvature > 0, the two features show very low interactivity.

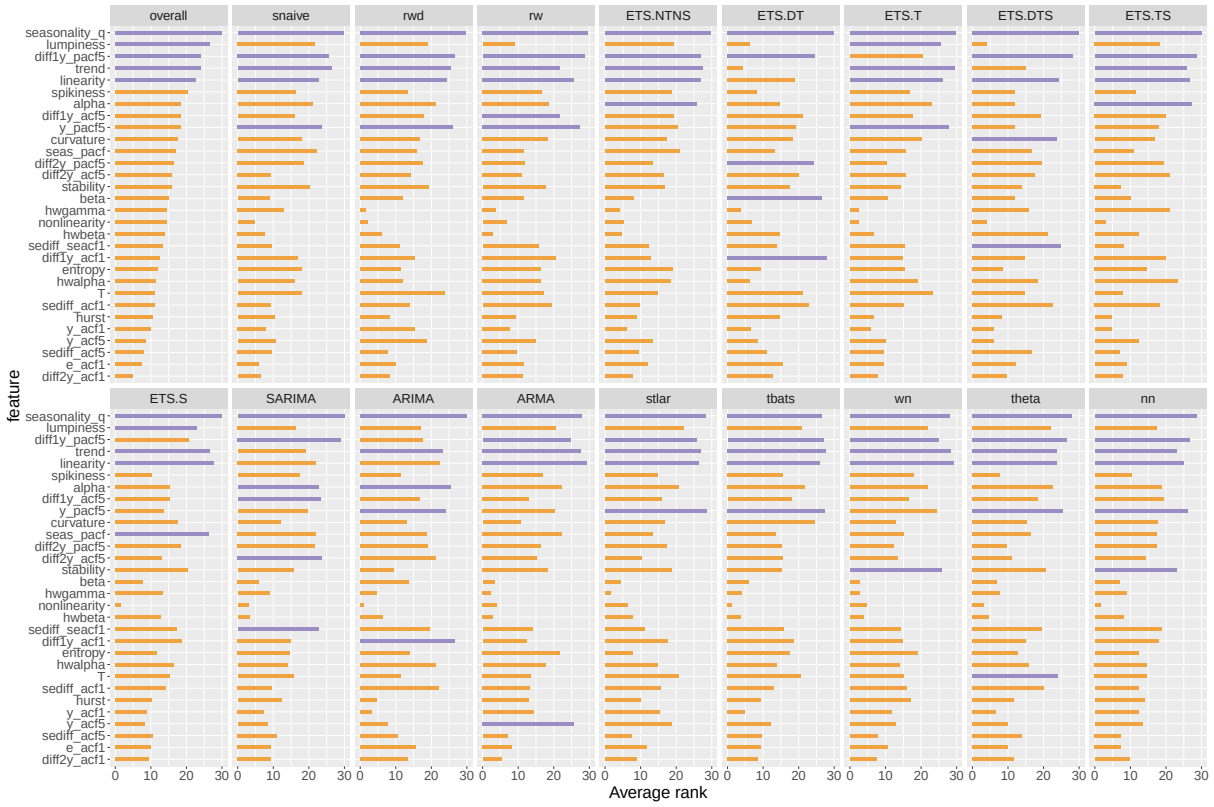

Figure 9: *Feature importance plot for quarterly series. Overall feature importance (top left plot) is evaluated based on two measures: i) permutation-based variable importance measure and ii) mean decrease in Gini coefficients. Class-specific feature importance is evaluated based on three measures: i) permutation-based variable importance, ii) PD-based variable importance measure, and iii) ICE-based variable importance measure. Longer bars indicate more important features. Top 5 overall features are highlighted in purple. Strength of seasonality appears to be the most important feature.*

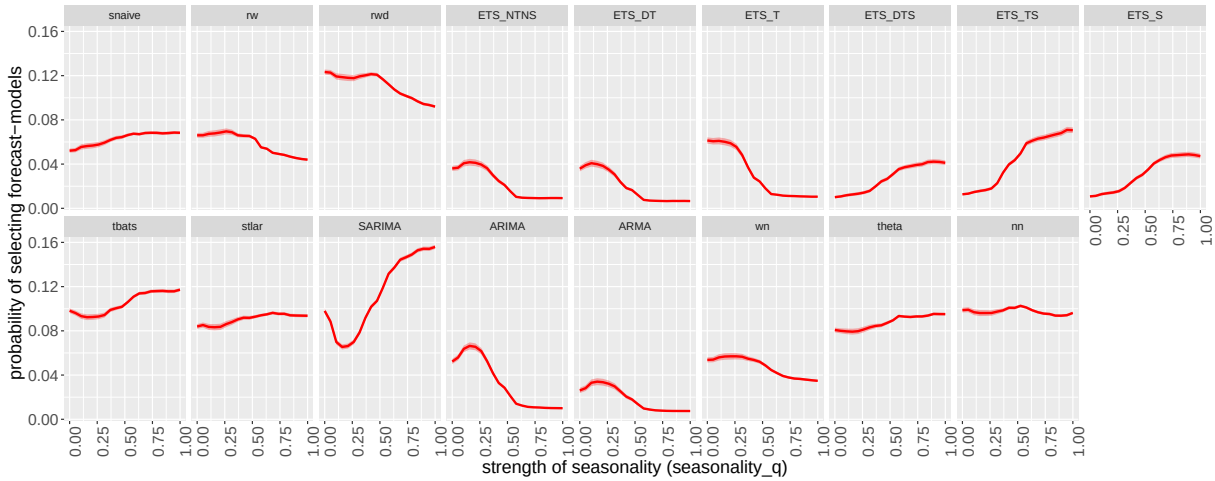

Figure 10: *Partial dependence plots for strength of seasonality (seasonality_q). The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting forecast models with seasonal components increases as seasonality increases.*

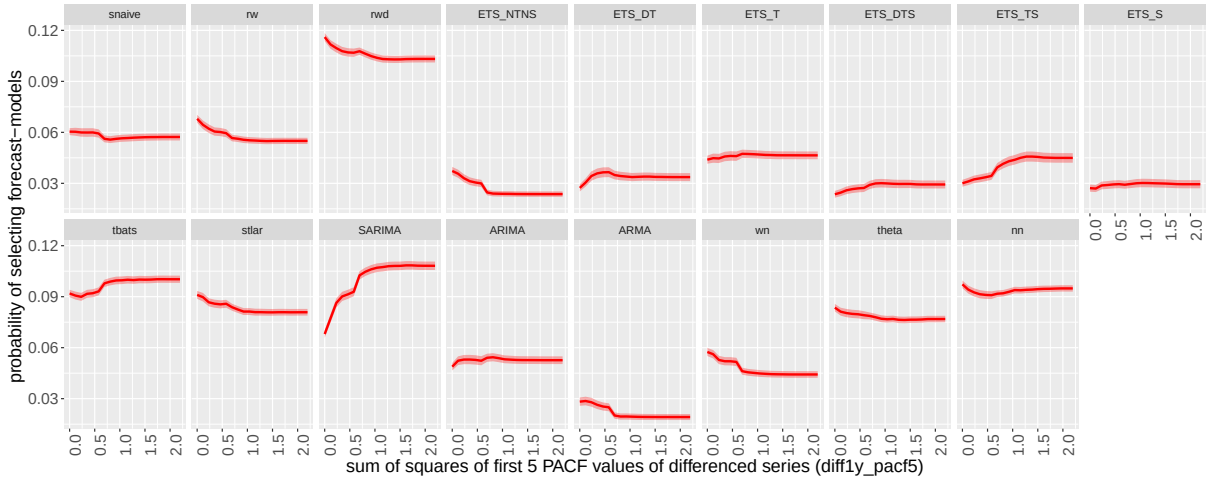

Figure 11: *Partial dependence plots for sum of squares of first 5 PACF values of differenced series (diff1y_pacf5). The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting SARIMA models drastically increases as diff1y_pacf5 increases*

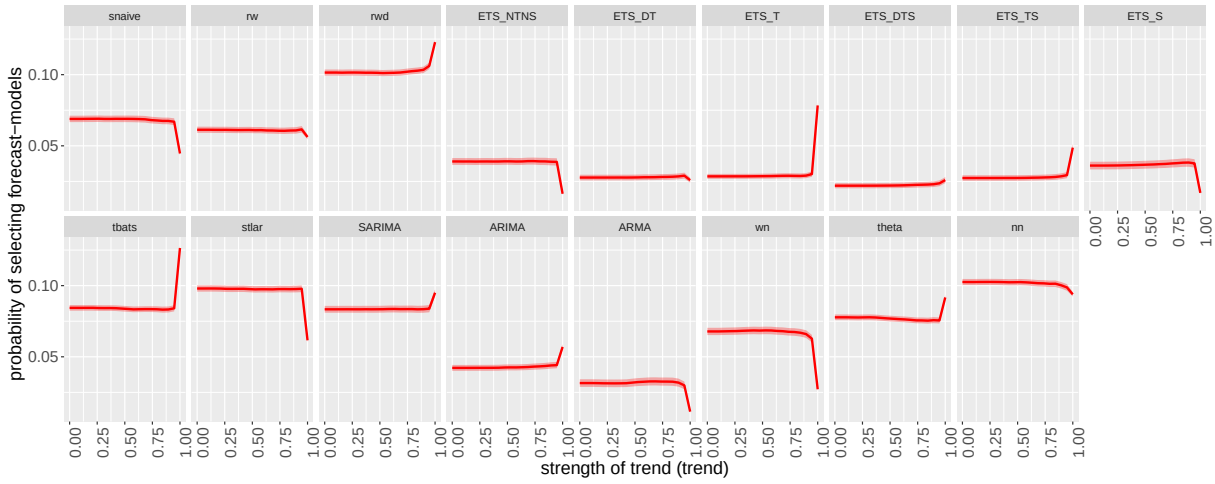

Figure 12: *Partial dependence plots for strength of trend (trend). The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. A very high value of strength of trend drastically influences the decision of selecting a forecast model.*

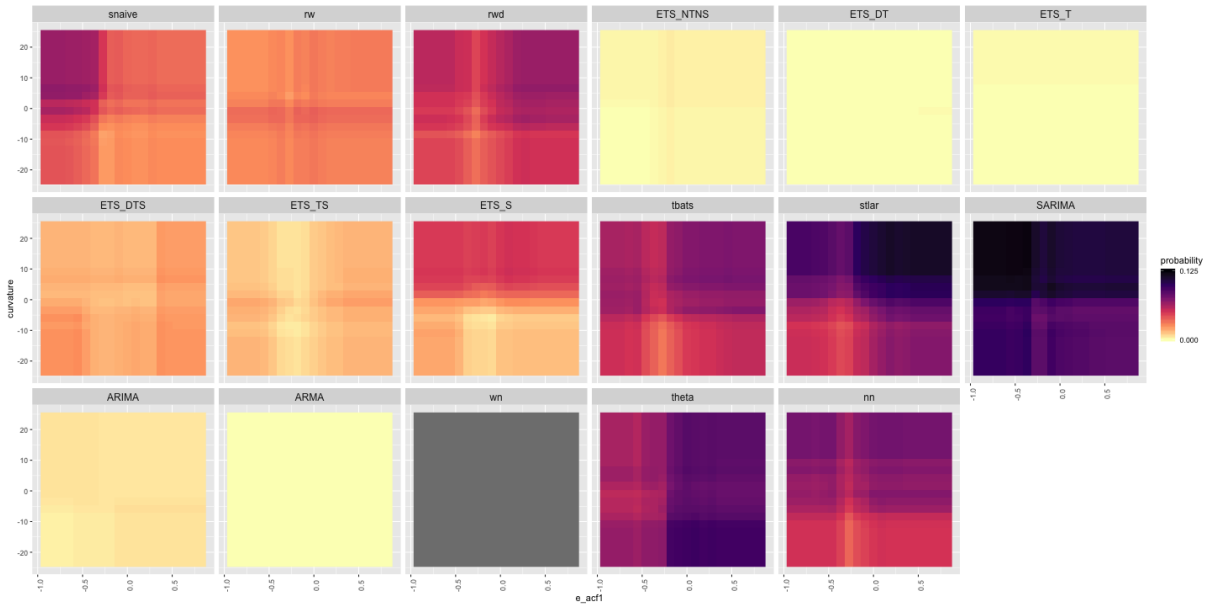

Figure 13: *Partial dependence of curvature and e_acf1 for quarterly series. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

7.3 Monthly series

Figure 14 shows the vote-matrix information for monthly series. Vote matrix information for monthly data depicted a similar pattern to quarterly data. For quarterly and monthly data, the same set of features and the class labels are used to train the model. Hence, this consistency between the results of the quarterly and the monthly series would provide evidence in support of the validity and trustability of the model.

Figure 15 shows the feature importance plots for monthly data. For both quarterly and monthly data strength of seasonality, trend, linearity and spikiness are the most important features across all categories. In the case of yearly series, low variable importance is assigned to both stability and length of the series. However, within quarterly and monthly data, a high variable importance is assigned to length of the series and stability. In addition to the strength of seasonality, the models handling seasonal components (snaive, SARIMA and all ETS models with seasonal components) assigned a high importance to the additional features related to seasonality such as ACF, PACF-based features related to seasonal lag and seasonally differenced series. Further, as expected, features calculated based on parameters estimated from ETS (A, A, A) ranked as important for the choice of ETS with damped trend and seasonal component and ETS with trend and seasonal component. One notable difference between the quarterly series and the monthly series is that for monthly data, the length of the series is ranked among the top five, especially in random walk with drift, random walk, ETS with seasonal and trend components, ETS with a seasonal component, SARIMA and ARIMA classes.

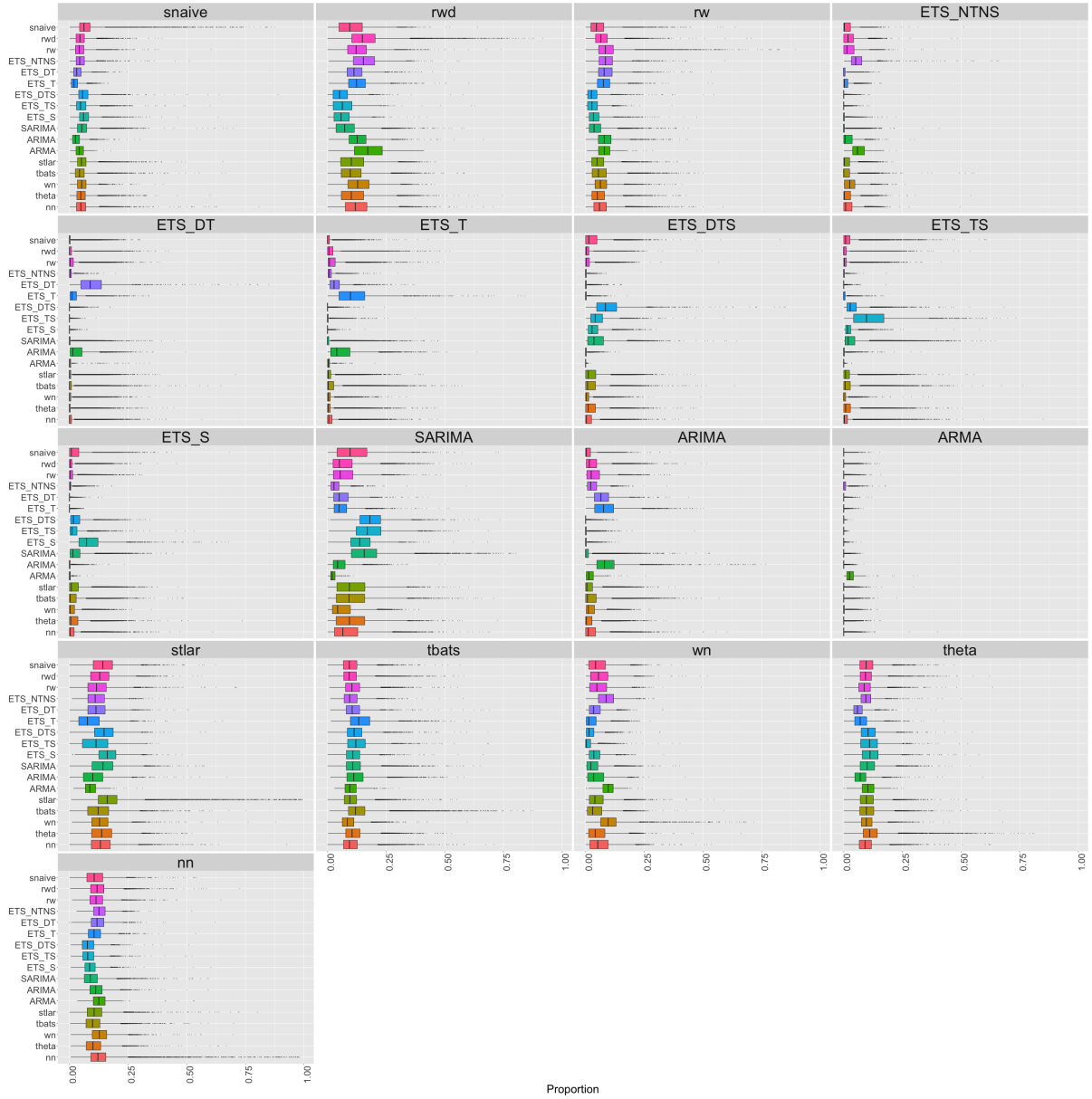

Figure 14: *Visualisation of the vote matrix based on OOB sample for monthly random forest. Each panel shows the predicted class from the random forest. The X-axis denotes the proportion of times each time series was classified to each class. The colours of boxplots correspond to class label of the best forecast model identified based on MASE and sMAPE. On each row, distribution of correctly classified class dominates, indicating a good classification of the meta-learner.*

The partial dependency curves of seasonality and trend of monthly data are very similar to the corresponding plots of quarterly data. Hence, these partial dependency curves are not repeated here. Figure 16 shows a partial dependency plot of linearity for monthly data. For non-seasonal forecast models the relationships are very similar to the corresponding observed curves in yearly data. Partial dependency curves of Figure 17 reveal highly parameterised models and models involving seasonal components such as (ETS_TS, ETS_DTS, ETS_S, SARIMA, tbats), which show an increasing relationship with length (T) up to some point. Figure 18 shows that partial

dependency plots correspond to the interaction between hwalpha and sediff_seacf1. Friedman's H-statistic (0.78) shows a very high level of interactivity between the features hwalpha and sediff_seacf1 with neural network class. In addition to wn class, the interaction between the two features is more useful in separating out the rwd and stlar from the rest, as they show high probability of selecting the corresponding models throughout the region. The tbats class shows low interactivity between the two features. The main effect of hwalpha dominates within tbats because of the constant colour of the horizontal bands. The classes snaive, rw, rwd, ETS_NTNS, stlar, theta and nn show varying degrees of colour strength bands, indicating that the probability of selecting the models changes according to the levels of both hwalpha and sediff_acf1.

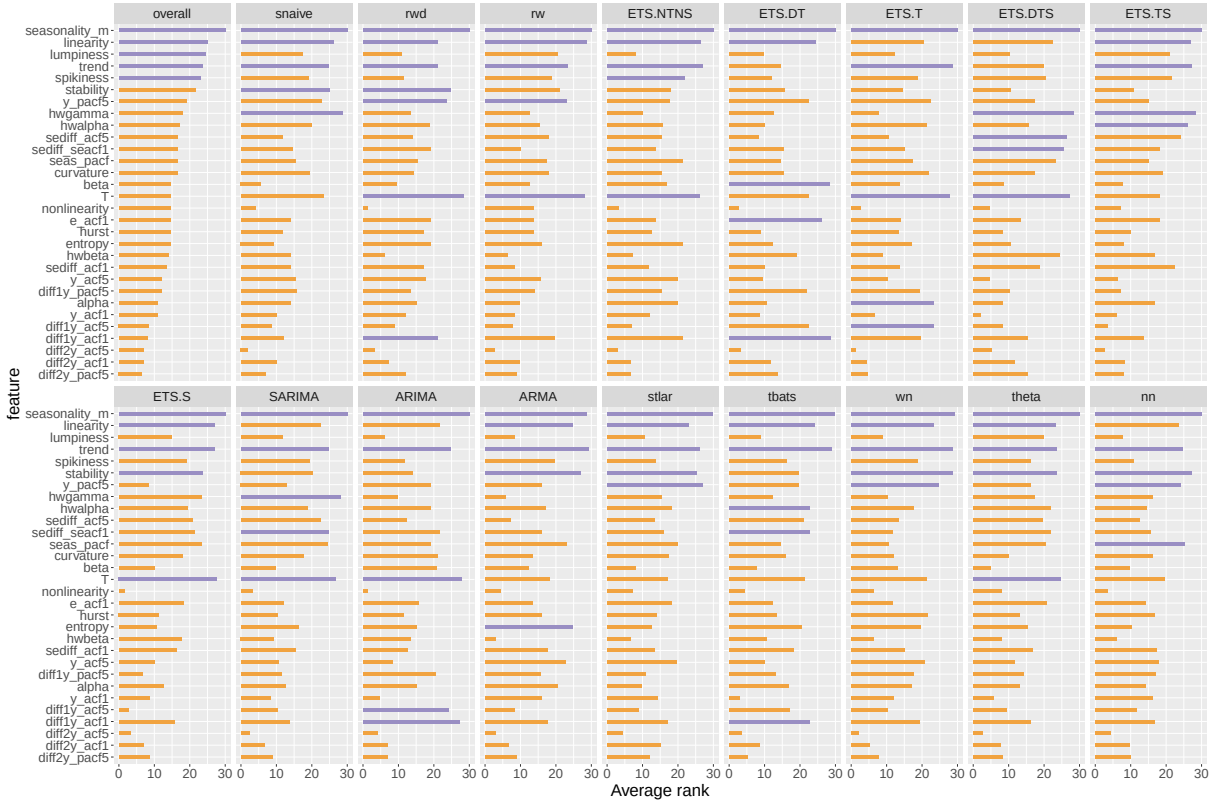

Figure 15: *Feature importance plot for monthly series. Overall feature importance (top left plot) is evaluated based on two measures: i) permutation-based variable importance measure and ii) mean decrease in Gini coefficients. Class-specific feature importance is evaluated based on three measures: i) permutation-based variable importance, ii) PD-based variable importance measure, and iii) ICE-based variable importance measure. Longer bars indicate more important features. Top 5 overall features are highlighted in purple. Strength of seasonality appears to be the most important feature.*

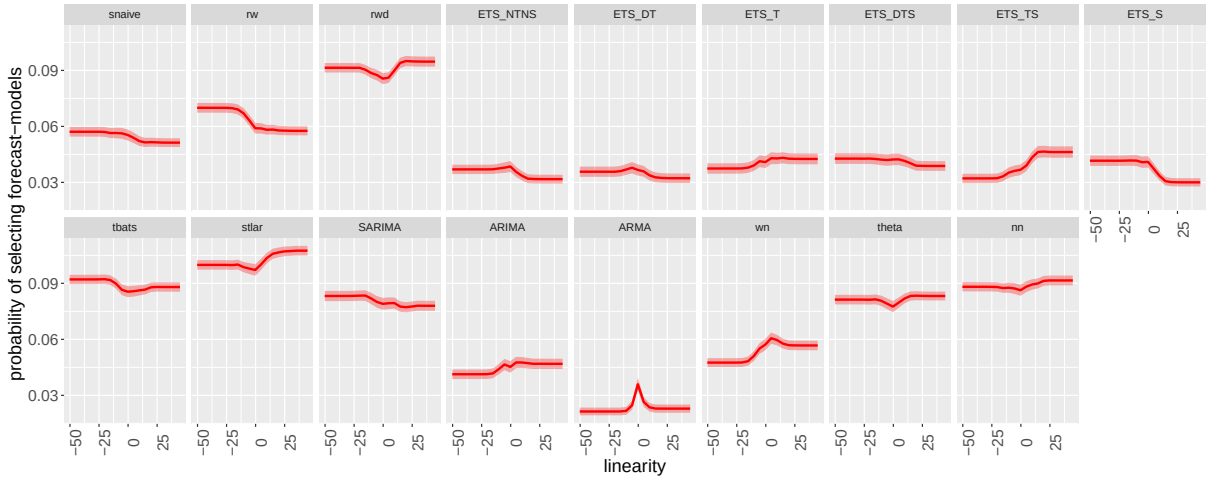

Figure 16: *Partial dependence plots for linearity. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. All curves show a turning point in the relationship around zero.*

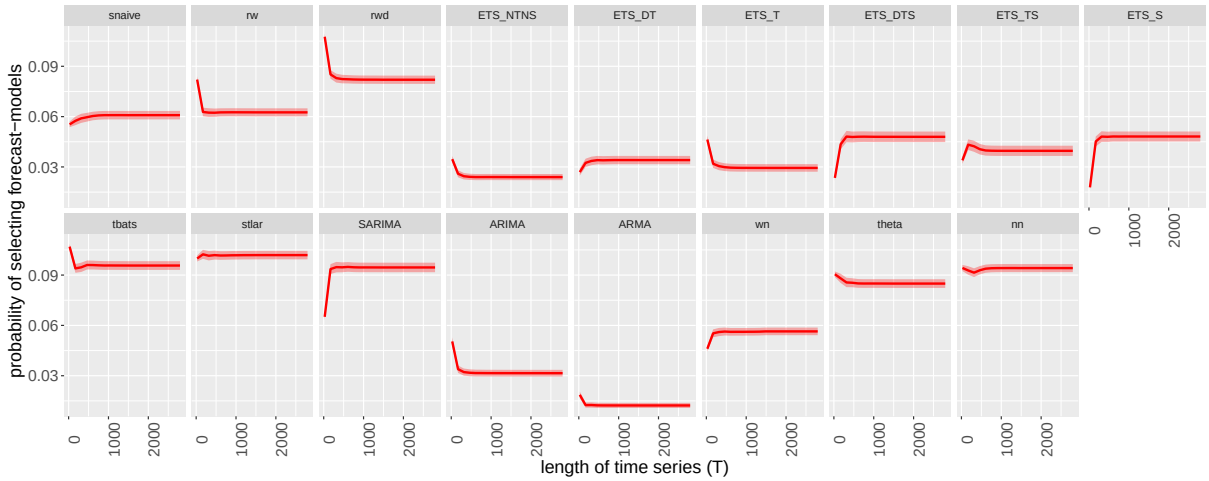

Figure 17: *Partial dependence plots for length of time series (T). The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class. Probability of selecting rw and rwd decreases as the length > 500.*

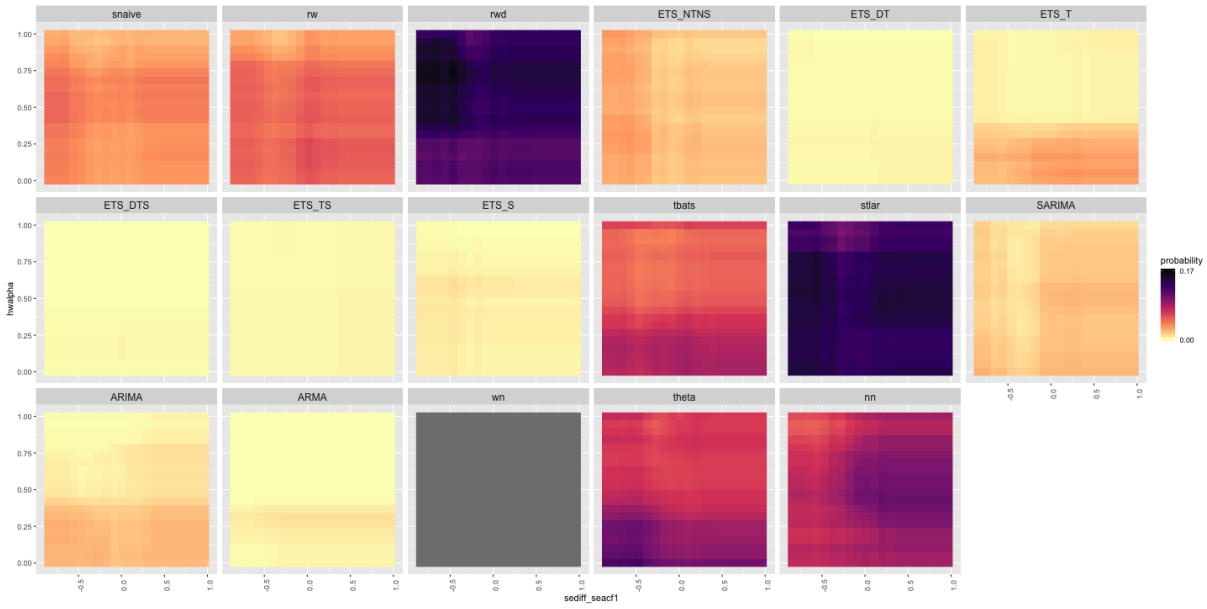

Figure 18: *Partial dependence of hwalpha and sediff_seacf1 for monthly data. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

7.4 Weekly series

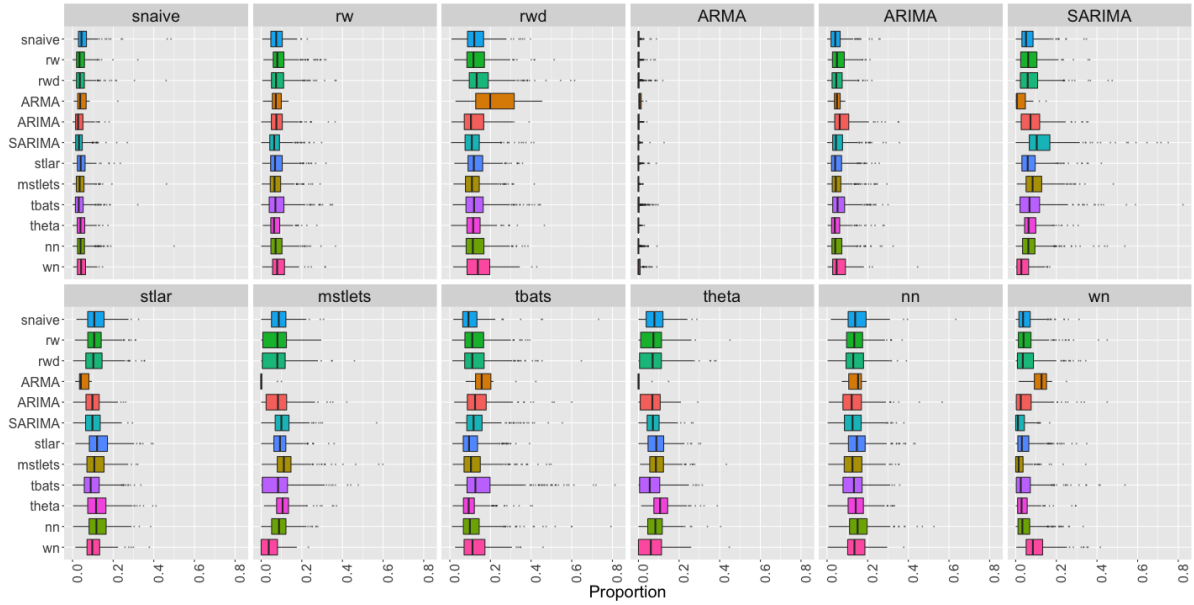

Figure 19: *Visualisation of the vote matrix based on OOB sample for weekly random forest. Each panel shows the predicted class from the random forest. The X-axis denotes the proportion of times each time series was classified to each class. The colours of boxplots correspond to class label of the best forecast model identified based on MASE and sMAPE. The models rwd, tbats and nn have a high chance of being selected.*

Figure 19 shows the vote matrix information for weekly data. Unlike yearly, quarterly and monthly data, theta method has a low chance of being selected. The rwd, tbats models and nn have a high chance of being selected. ARMA class shows some unusual behaviour within some

categories because of class imbalance ratio; it contains a smaller number of observations in the training set.

According to the results of Figure 20, spikiness, linearity, trend, strength of seasonality, stability and lumpiness have been assigned a high importance. This is similar to the results of yearly, quarterly and monthly data. The length of series has been selected among the top five by mstlets, tbats, theta and nn models. The partial dependence plots for seasonality, trend and linearity are very similar to the patterns observed in quarterly and monthly data. Hence, we have not repeated the results. Instead, we have plotted the partial dependence plot for spikiness. According to Figure 21, the probability of selecting snaive, random walk, neural network and white noise increases as the spikiness increases, while other forecast model classes show the opposite. According to Friedman's H-statistic, curvature and linearity is the most commonly appearing two-way interactivity. The associated two-way partial dependency plots are shown in Figure 22. The interactivity pattern of ARMA is not visible between 0-0.15 scale as ARMA models have low chance of being selected with the weekly series. A unique pattern of interactivity could be observed within snaive, rw, SARIMA, stlar, mstlets, tbats, theta and nn. Within rwd class we can see the main effect of linearity dominating; however, the strength of colour varies according to the level of curvature.

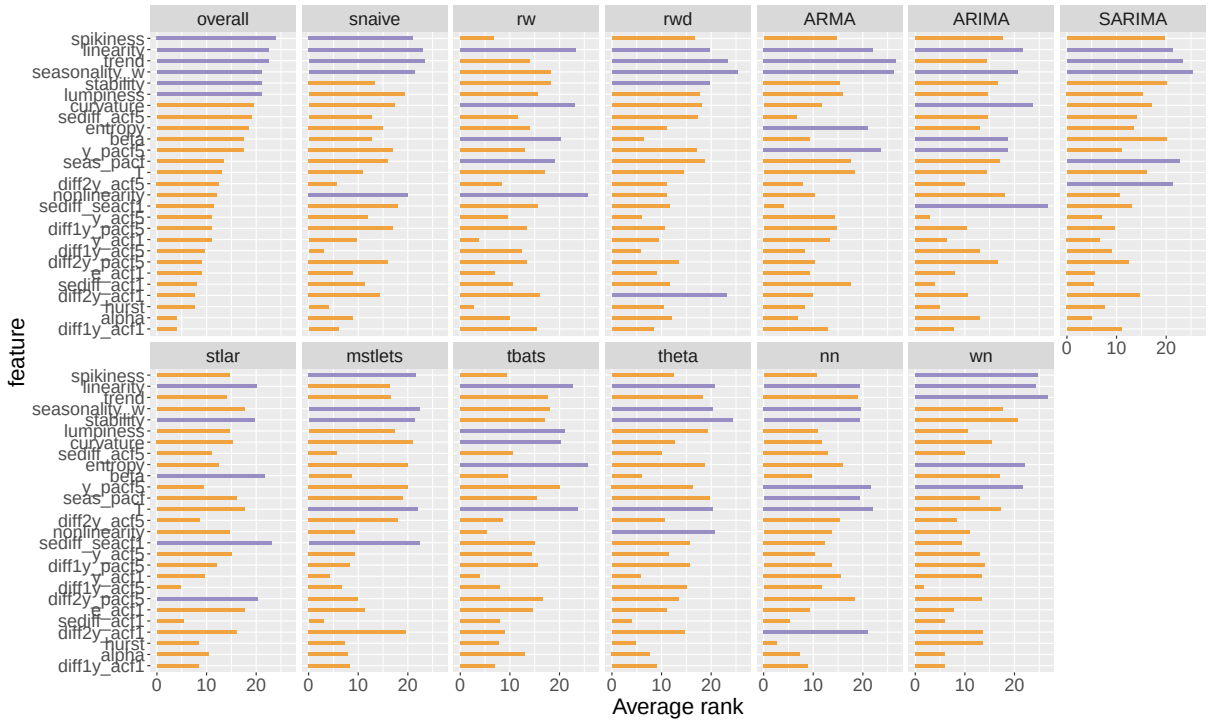

Figure 20: *Feature importance plot for weekly data. Permutation-based VI measure and mean decrease in Gini coefficient are used to evaluate overall feature importance. Class-specific feature importance is evaluated based on the three measures:i) permutation-based VI, ii) PD-based VI measure, and iii) ICE-based VI measure. Longer bars indicate more important features. Top 5 features are highlighted in purple.*

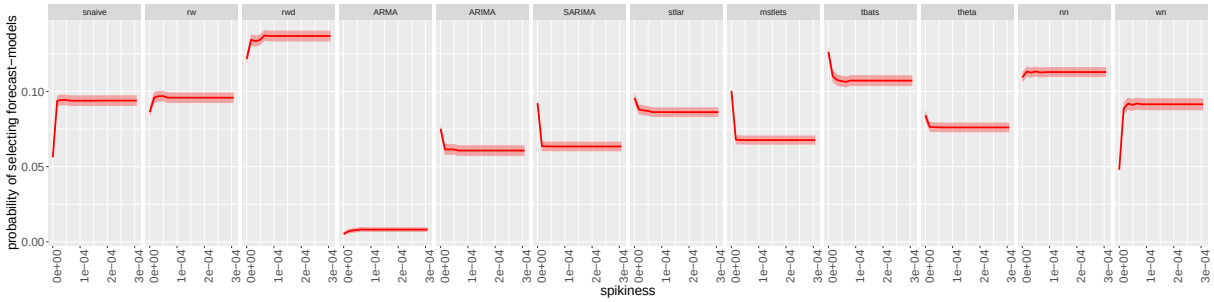

Figure 21: *Partial dependence plots for spikiness. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

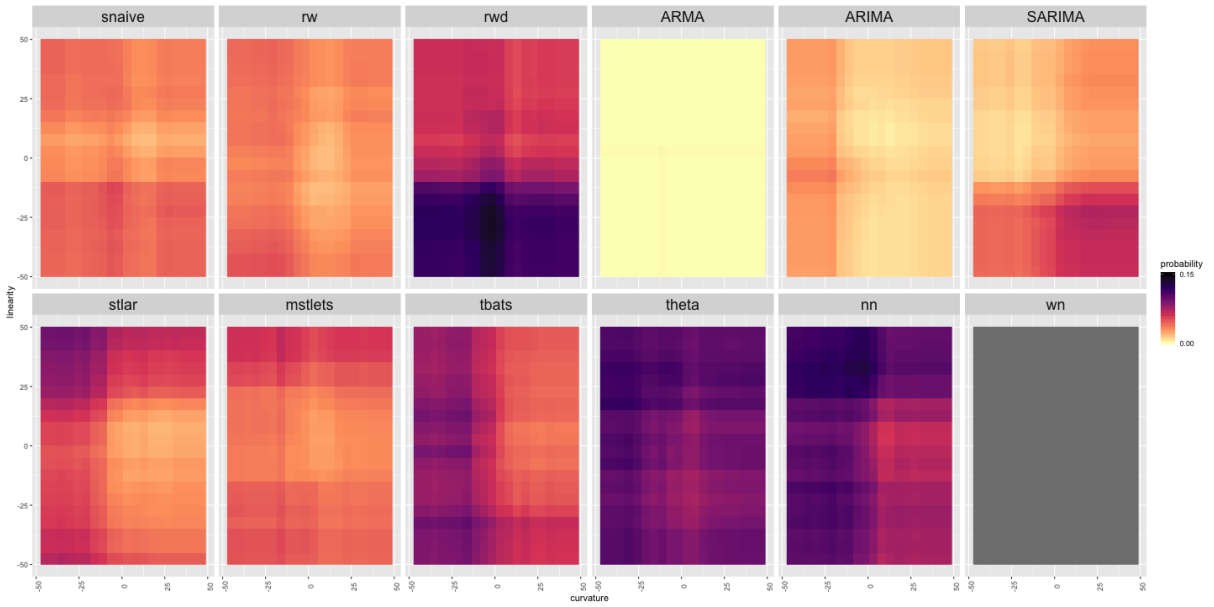

Figure 22: *Partial dependence of linearity and curvature for weekly data. Dark regions show the high probability of belonging to the corresponding class shown in the plot title.*

7.5 Daily series

Figure 23 shows the vote matrix information for daily series. According to Figure 23, the distributions corresponding to observations that have been correctly classified dominate the top for daily data. However, within daily series there are few observations that have been incorrectly classified to tbats class with very high probabilities. In general, neural network models have a higher chance of being selected for daily time series.

The variable importance graph for daily data is shown in Figure 24. The most important features for daily time series are, strength of seasonality corresponds to the weekly seasonality (measured by seasonal_w), stability, trend, lumpiness and linearity. Further, the length of the series is important in determining random walk, random walk with drift, mstlarima, mstlets, stlar, theta and nn classes.

In Figure 25 we can see that the probability of selecting white noise models increases as stability increases. According to the partial dependency curves of length (T), Figure 26, shorter series tend to select random walk with drift models while the probability of selecting snaive, mstlarima and mstlets models increases as the length of the series increases. Neural network models show a non-monotonic relationship with the length of the series (T). For daily time series, stability and strength of seasonality (seasonality_w) shows high levels of two-way interactivity within many classes (Figure 27). The classes theta and nn show high levels of interactivity between seasonal_w and stability. Within theta and nn the separation of quadrants is easily seen. For example, with low values of seasonal_w and low stability, nn models have a high chance of being selected, while with high values of seasonal_w and stability, nn models have a low chance of being selected. The opposite relation can be observed within rw, mstlarima, tbats and theta classes.

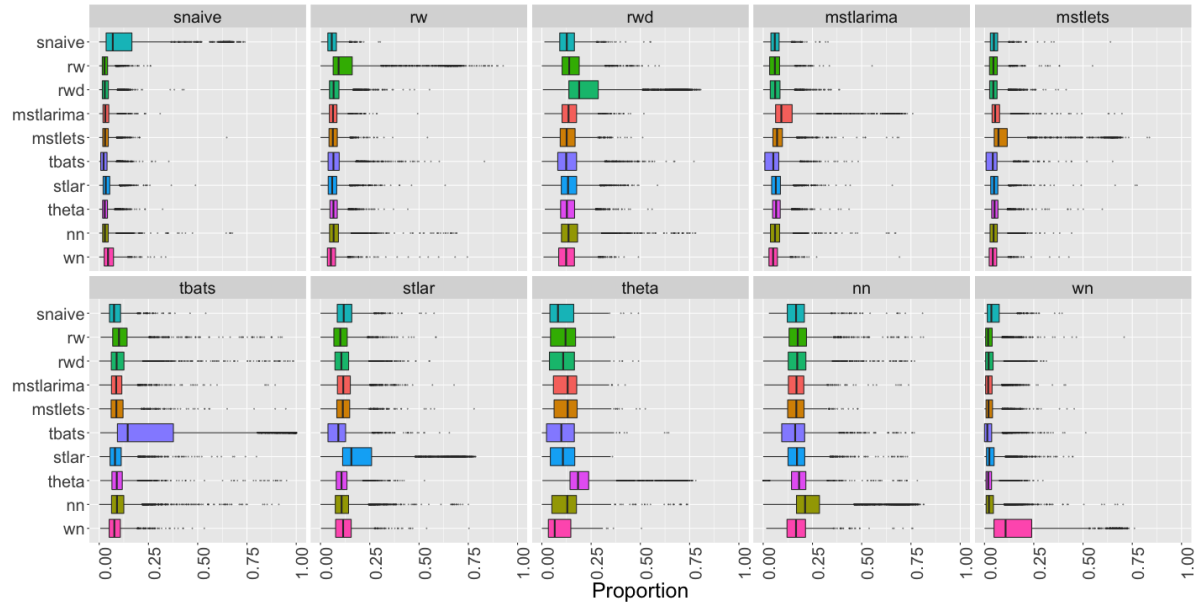

Figure 23: *Visualisation of the vote matrix based on OOB sample for daily random forest. Each panel shows the predicted class from the random forest. The X-axis denotes the proportion of times each time series was classified to each class. The colours of boxplots correspond to class label of the best forecast model identified based on MASE and sMAPE. The models rwd, tbats and nn have a high chance of being selected.*

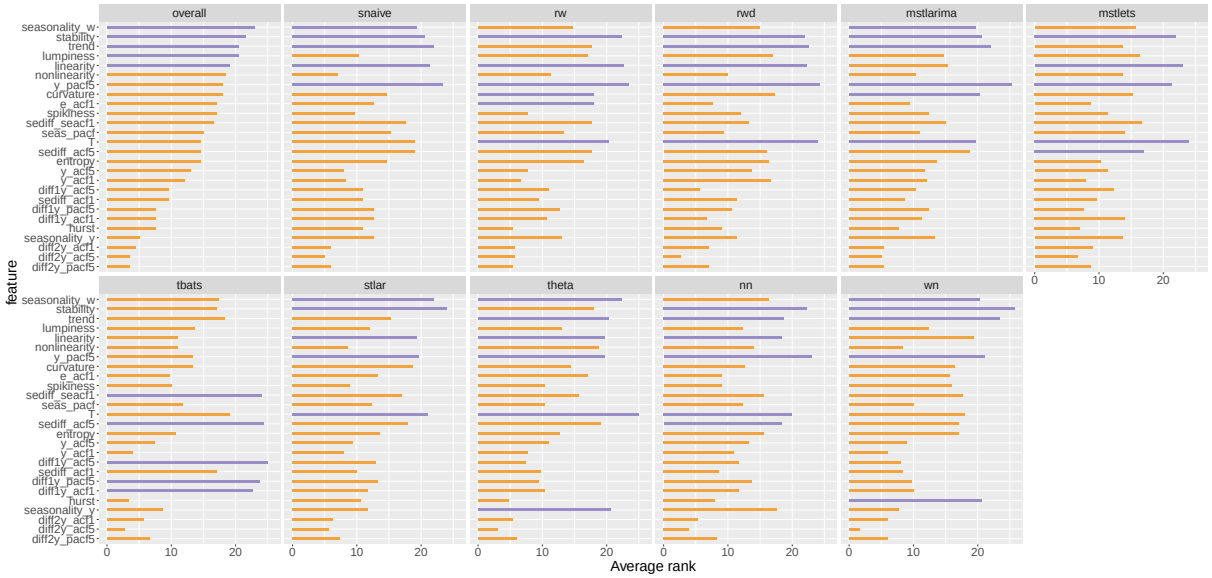

Figure 24: *Feature importance plot for daily data. Permutation-based VI measure and mean decrease in Gini coefficients are used to evaluate overall feature importance. Class-specific feature importance is evaluated based on the three measures: i) permutation-based VI, ii) PD-based VI measure, and iii) ICE-based VI measure. Longer bars indicate more important features. Top 5 features are highlighted in purple.*

Figure 25: *Partial dependence plots for stability. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

Figure 26: *Partial dependence plots for length. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

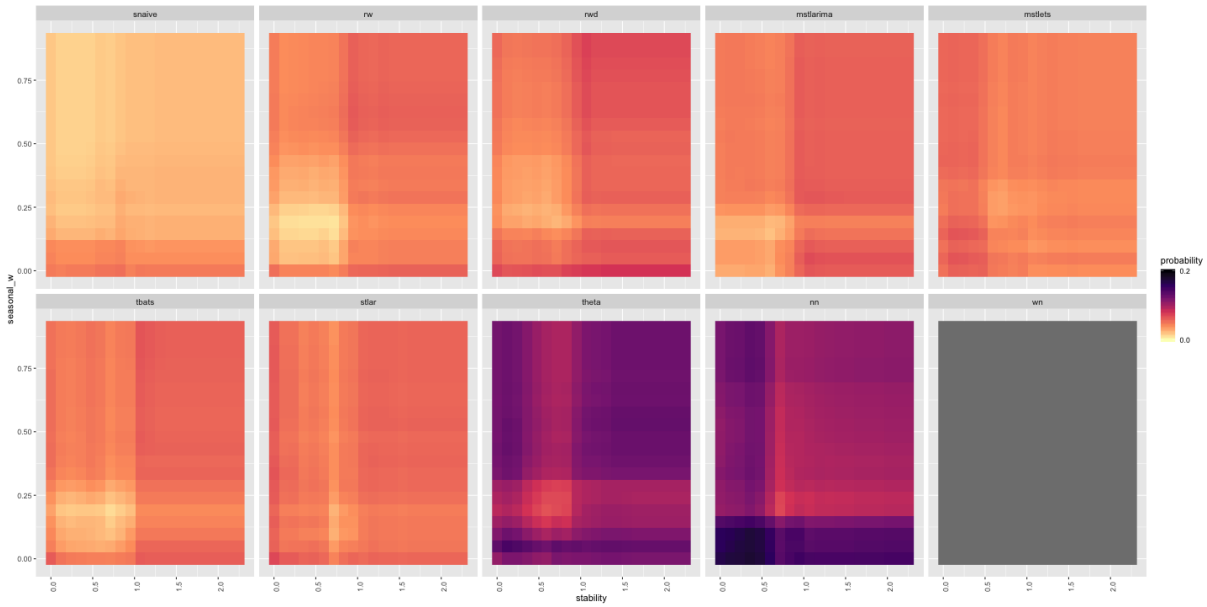

Figure 27: *Partial dependence of seasonal_w and stability for daily data. Dark regions show the high probability of belonging to the corresponding class shown in the plot title. Random walk (rw) and nn show opposite patterns of interactivity between seasonal_w and stability.*

7.6 Hourly series

Figure 28 shows the vote matrix information for hourly data. Overall, for hourly series random walk with drift, tbats and neural network models have a high chance of being selected. Further, it is important to note that within nn all distributions are located away from zero, which indicates all hourly series have been assigned a non-zero probability of being selected to neural network class.

The variable importance graph for hourly data is shown in Figure 29. According to Figure 29, the strength of daily seasonality (measured by seasonal_d) appears to be more important than the strength of weekly seasonality (measured by seasonal_w). However, the strength of weekly seasonality also seems to be one of the most important features for classes snaive, random walk, mstlarima, and tbats. Further, entropy, linearity, sum of squares of first five coefficients of PACF, curvature, trend, spikiness and stability were found to be the most important features in determining the best forecasting method for hourly time series. Only the snaive category ranked T among the top five for hourly time series.

The partial dependency plots of the strength of seasonalities are shown in Figure 30. According to Figure 30, the probability of selecting random walk, random walk with drift, theta and white noise process decreases with a higher value of strength of daily seasonality. On the other hand, the probability of the selecting random walk model increases as the strength of weekly seasonality increases. The partial dependency plots of entropy are shown in Figure 31. The

entropy measures forecastability of time series. We can see that the probability of selecting random walk with drift and tbats decreases as the entropy increases. This is consistent with our theoretical expectations. In other words, if the series has a clear trend or a clear trigonometric pattern, the forecastability of the time series is high and the entropy is low. For highly trended series the random walk with drift model is suitable, while for a series with trigonometric pattern the tbats model is suitable.

Friedman's H-Statistic shows high levels of interactivity between sediff_seacf1 and linearity across all classes. The associated two-way partial dependency plot is shown in Figure 32. According to the Figure 32, we can see there is a unique pattern of interactivity existing within each class. Within rw, rwd and mstlarima we can see a separation between the lower half and the upper half due to the effect of sediff_seacf1. However, the colours of the bands are not uniform across the bands, which indicates that the probability of selecting the models changes according to the levels of linearity. The two-way partial dependency plots show rwd has a high chance of being selected with series with low sediff_seacf1; this is due to the over differencing of the series.

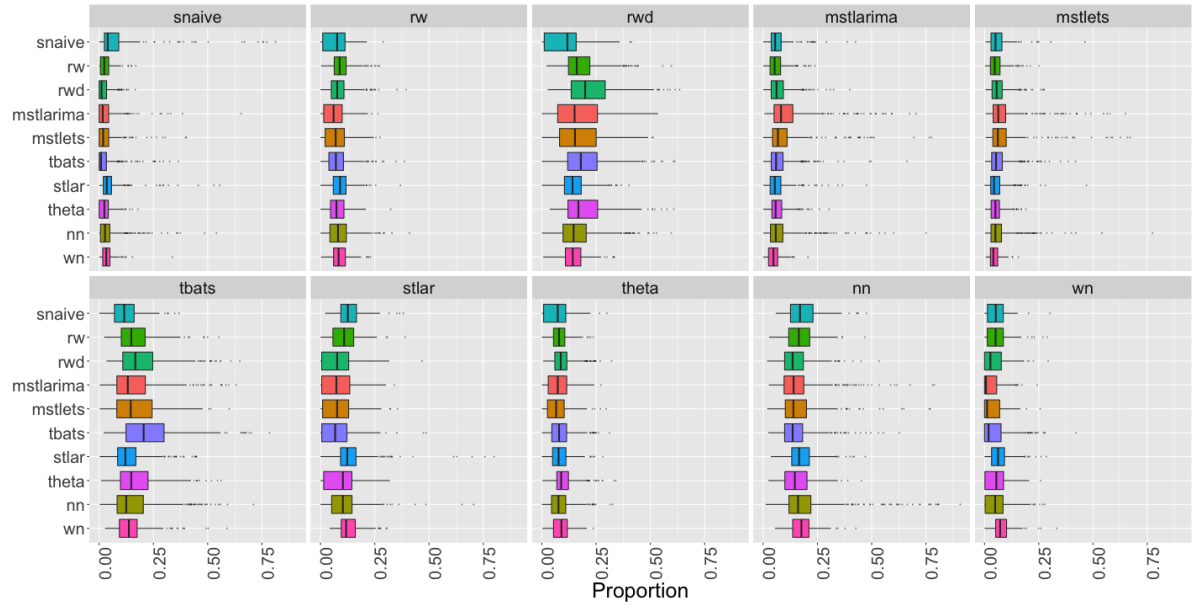

Figure 28: *Visualisation of the vote matrix based on OOB sample for hourly random forest. Each panel shows the predicted class from the random forest. The X-axis denotes the proportion of times each time series was classified to each class. The colours of boxplots correspond to class label of the best forecast model identified based on MASE and sMAPE. The models rwd, tbats and nn have a high chance of being selected.*

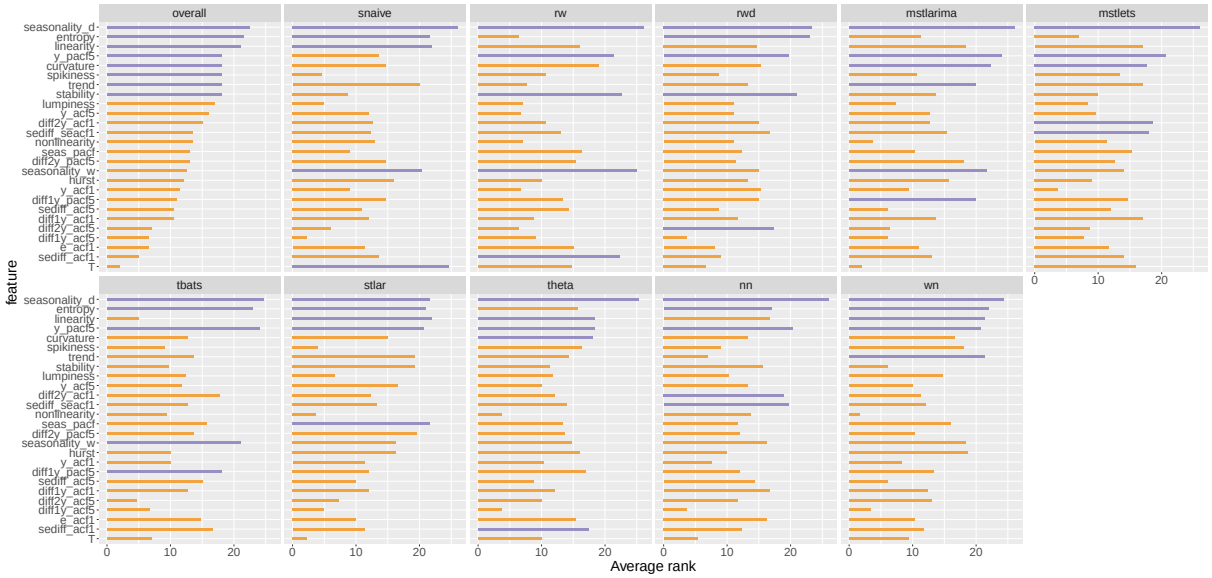

Figure 29: *Feature importance plot hourly series. Permutation-based VI measure and mean decrease in Gini coefficients are used to evaluate overall feature importance. Class-specific feature importance is evaluated based on the three measures: i) permutation-based VI, ii) PD-based VI measure, and iii) ICE-based VI measure. Longer bars indicate more important features. Top 5 features are highlighted in purple.*

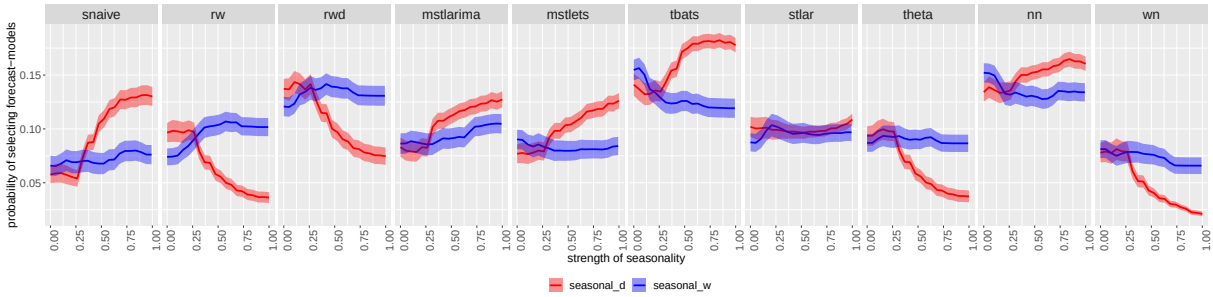

Figure 30: *Partial dependence plots for strength of seasonality. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

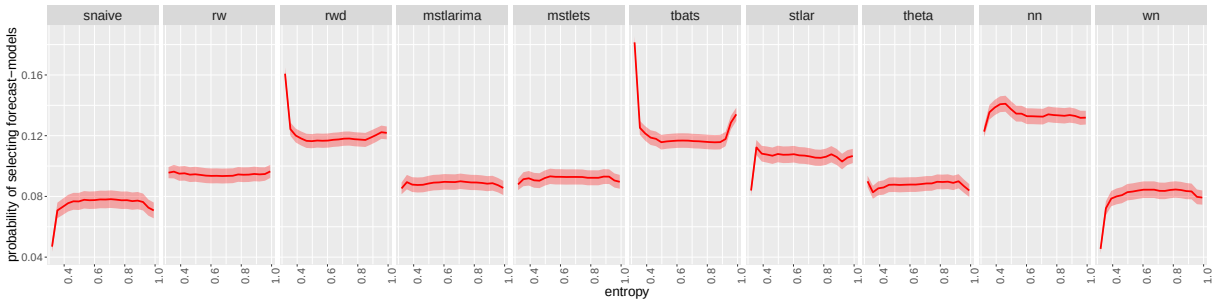

Figure 31: *Partial dependence plots for entropy. The shading shows the 95% confidence intervals. Y-axis denotes the probability of belonging to the corresponding class.*

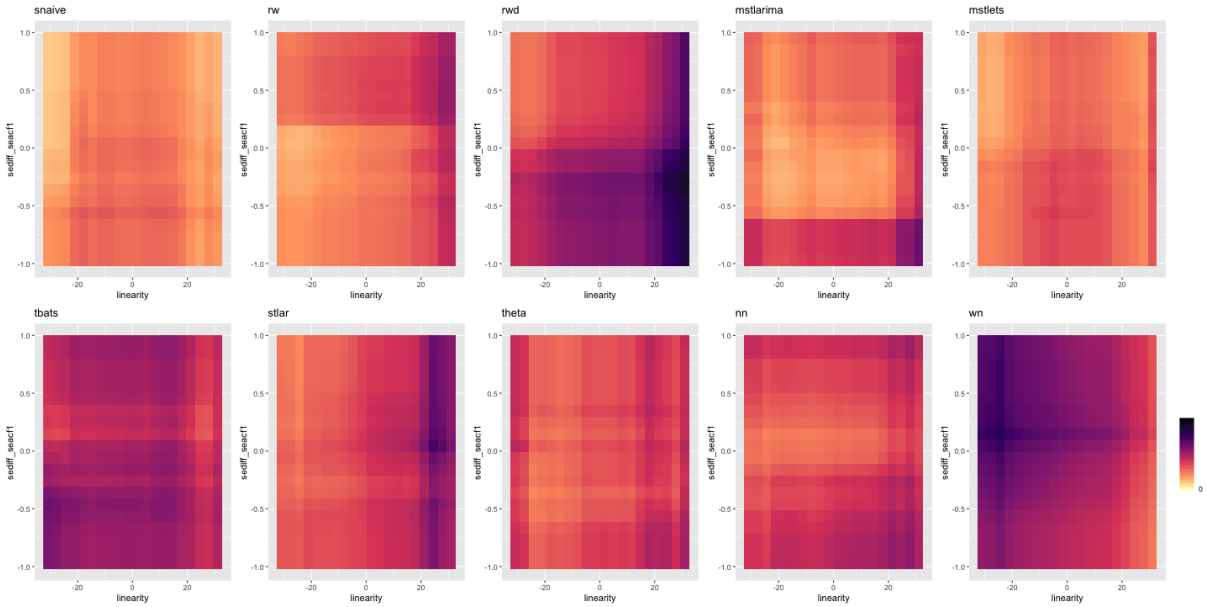

Figure 32: *Partial dependence of sediff_seacf1 and linearity for hourly data. Dark regions show the high probability of belonging to the corresponding class shown in the plot title. Random walk (rw) and random walk with drift (rwd) show opposite pattern of interactivity between sediff_seacf1 and linearity.*

7.7 Local interpretable model-agnostic explanations

We now illustrate how the LIME approach can be used to zoom into local regions of the data to identify which features contribute most to the classification of a specific instance. For the illustration we select four different time series classified with high probability. Figure 33 shows the feature contribution for the instances highlighted on the PCA-space of quarterly series. We can see how the strength of seasonality influences the FFORMS framework to select different types of seasonal forecast models. For example, the SARIMA model is selected when the seasonality varies between 0.579 and 0.787 (case 1), the ETS-seasonal model is selected when the strength of seasonality is greater than 0.787 (case 4), random walk with drift when the seasonality is lower than 0.579 (case 2) and for the highly trended and seasonal series (strength of seasonality > 0.895) the ETS model with a trend and seasonal component is selected (case 3). Further, high value of diff1y_acf5 supports the selection of SARIMA for case 1; moderate value of diff1y_acf5 supports the selection of ETS-seasonal for case 4. Similarly, we can explore the reasons for other instances in all frequency categories. From the LIME approach we can gain insight into the local neighbourhood characteristics that lead to the choice of a particular neighbourhood over alternative destinations.

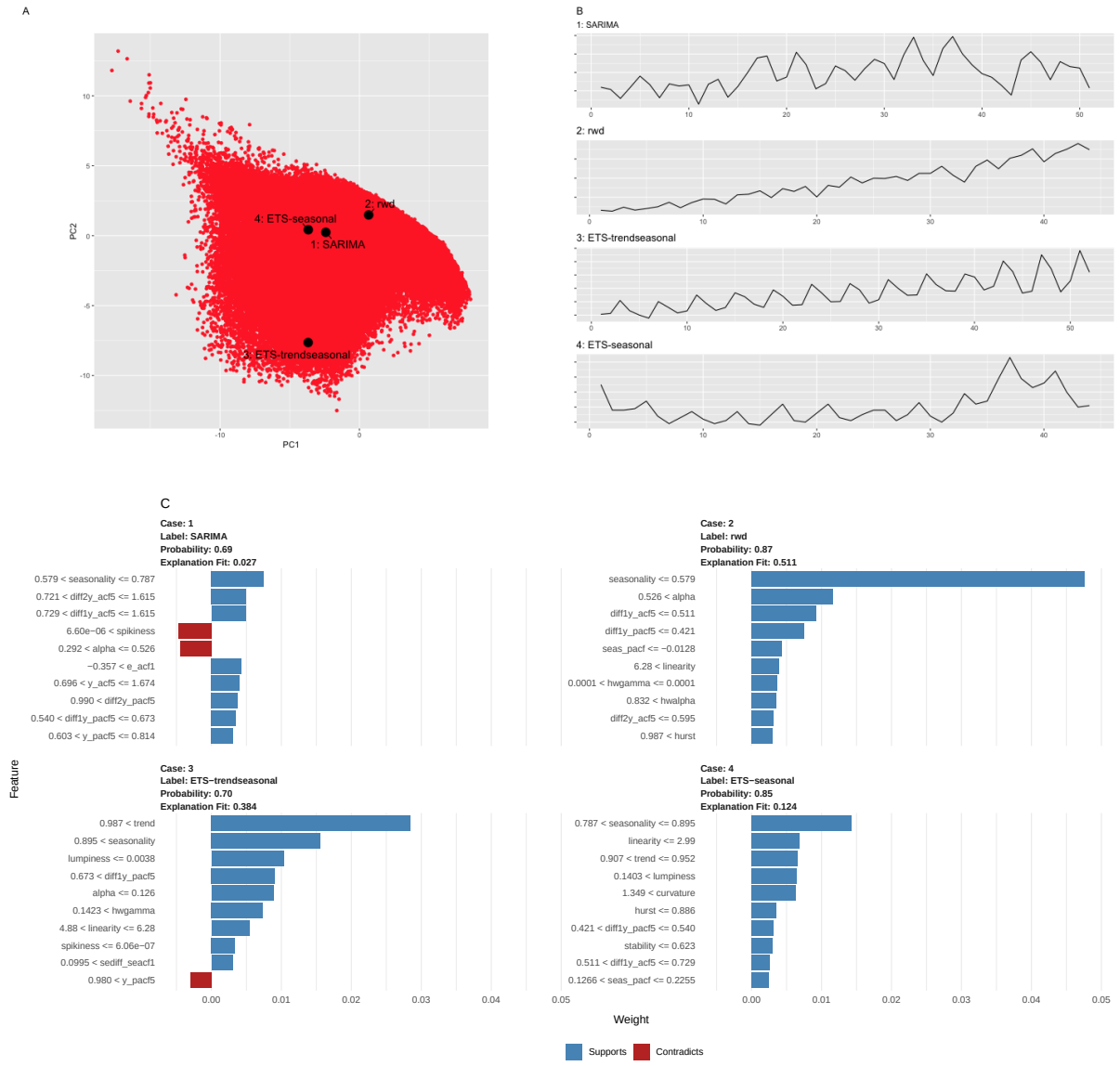

Figure 33: *Panel A: Distribution of quarterly time series in the PCA space. Panel B: Time series corresponds to the highlighted points in the PCA space. Panel C: Local interpretable model-agnostic explanations for four selected quarterly time series. Features denoted with blue colour are supporting features for an outcome label and length of the bar is proportional to the weight of a feature.*

8 Discussion and Conclusions

Features of time series are useful in identifying suitable models for forecasting. This paper explores the role of features in forecast model selection under the hood of the FFORMS framework. First, we evaluate the FFORMS prediction performance using the M4 competition data. Then we explore the dissimilarities and similarities of forecast models (class labels) learned by the FFORMS framework based on the vote-matrix information of the random forest. Next, we explore **which** features are the most important for the choice of the FFORMS framework, and **where** they are most important; for the overall classification process or within a specific class of forecast models or a set of multiple classes of forecast models. Partial dependency plots are

then used to visualise **when** and **how** these features link with the prediction outcome of the FFORMS framework. Finally, we explore **when** and **how strongly** features interact with other features using Friedman's H-Statistic.

Features such as the strength of trend and strength of seasonality rank top within each class across all frequency categories (see Table 5). In addition, linearity, curvature and spikiness also rank among the top five within most of the classes. Within most of the classes at least one feature related to autocorrelation and partial autocorrelation coefficients ranks among the top five. This confirms that the information regarding the correlation structure of the time series is essential for the choice of model selection. The length of time series also appeared to be important in selecting simple forecast models such as snaive, naive and random walk with drift. Partial dependency plots of length show that short time series tend to select simple forecast models such as snaive, naive and random walk with drift while more parametrised forecast models such as SARIMA, ETS with trend and seasonal component, the neural network approach and forecast models handling multiple seasonal components are often selected with lengthy series. For all features, the displayed relationships of partial dependency plots are consistent with the domain knowledge expectations. This is an important aspect in encouraging people to trust and use the proposed framework effectively. However, several features are used to build the framework with comparable contributions, and thus all individual contributions are small. The observed two-way interactions do not completely change the individual relationship of features with the predicted outcome, but they do change the probability of selecting the models. Further, the LIME approach is used to explore the reasons behind each individual prediction. Exploration of other local interpretability methods is a direction for future research. This is useful for understanding the reasons behind series being classified with very high probability or very low probability. It also helps to increase trust in the framework because if people understand the reasons behind the results, they can use their prior knowledge about the application domain to decide whether to accept (trust) or reject the prediction outcome. The results of this help to refine the picture of the relationship between features and the choice of forecast model, which is particularly valuable for ongoing research in the field of feature-based time series analysis.

Exploring of the conditions learned by the FFORMS framework also supports practitioners to make a good educated guess about suitable forecast models for a given problem. Further, the results of this study are useful in identifying new ways to improve forecasting accuracy by capturing different features of time series.

Appendix A: Interaction effects

Figure 34-Figure 39 show how strongly each feature interacts with any other feature in the forest for yearly, quarterly, monthly, weekly, daily and hourly series. On each panel the top five interacting features are highlighted in purple.

According to Figure 3, which shows the variable importance plots for yearly data, the feature linearity is assigned a very high variable importance within rwd and ARMA, but according to Figure 34, linearity shows very low interactivity within rw and rwd. This means that linearity is more important on its own than with its interaction effect. Figure 6 further confirms this as the partial dependency curves corresponding to rwd and ARMA drastically change as linearity varies. According to Figure 35, quarterly data strength of seasonality shows relatively low interactivity, indicating the importance of the main effect of seasonality for the classification process. Further, the length of the time series shows a high level of interactivity within the random walk class. According to Figure 36 and Figure 37, in the process of classifying monthly and weekly series the number of features showing a high level (>0.5) of interactivity within classes is relatively low. This reflects the importance of the individual effect of features when selecting forecast models for monthly and weekly series. Weekly data FFORMS frameworks show a high level of interactivity of features within ARMA. This is due to the class imbalance. Weekly series reference data sets contained a very small number of ARMA labelled series. Hence, features interact highly to separate ARMA from the rest. Similar to the results of weekly and monthly data, according to Figure 38, the number of features that show high level of interactivity (> 0.7) with other features is relatively low.

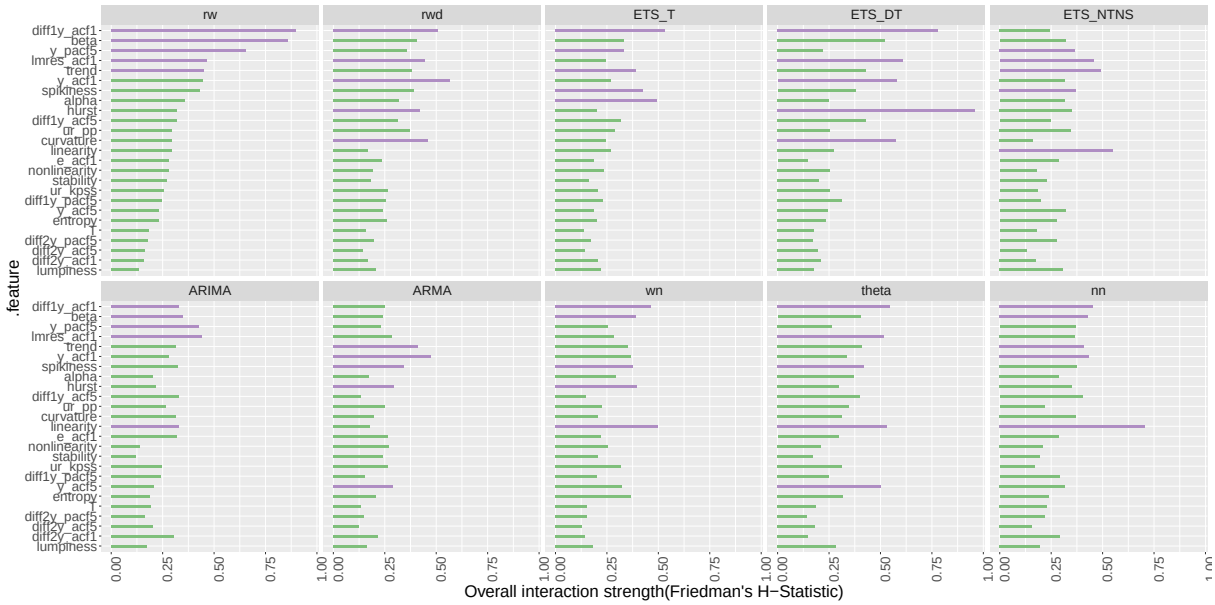

Figure 34: *The interaction strength (Friedman's H-Statistic) for each feature with any other feature for yearly forest. The Hurst exponent within ETS_DT has the highest interaction effect.*

Figure 35: *The interaction strength (Friedman's H-Statistic) for each feature with any other feature for quarterly forest. Top 5 interacting features are highlighted in purple. Strength of seasonality shows an extremely low interactivity within SARIMA, ETS_DT and ETS_DTS.*

Figure 36: *The interaction strength (Friedman's H-Statistic) for each feature with any other feature for monthly forest. Top 5 interacting features are highlighted in purple. Curvature shows highest level of interactivity within nn*

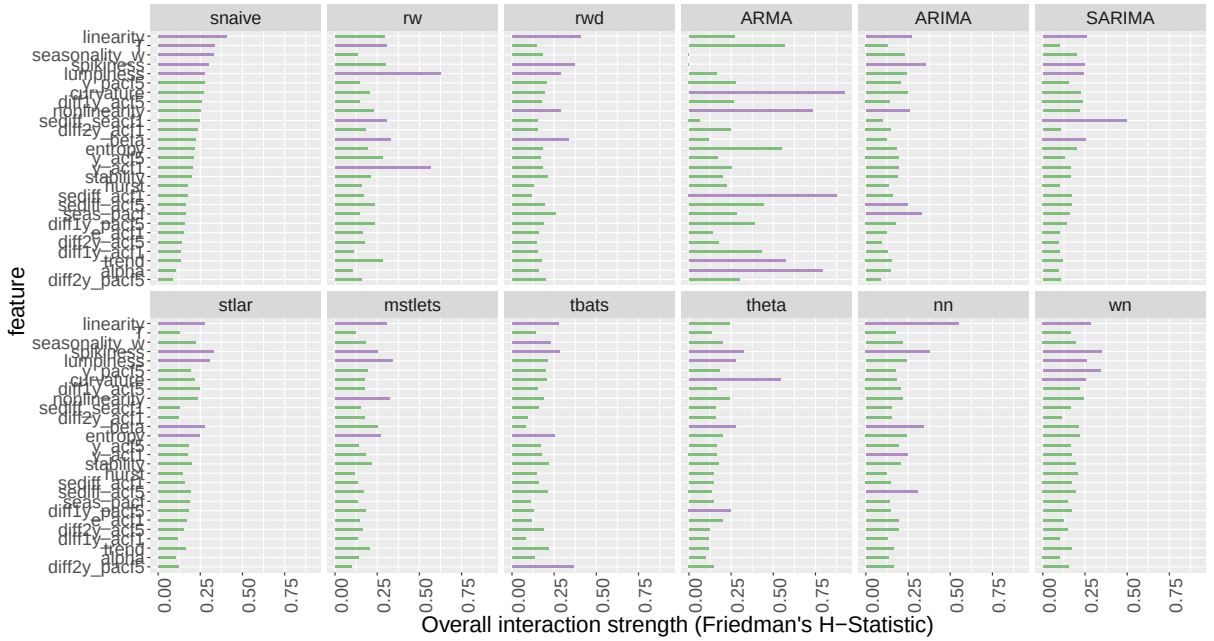

Figure 37: *The interaction strength (Friedman's H-Statistic) for each feature with any other feature for weekly forest. Overall, the interaction effects between the features are very weak.*

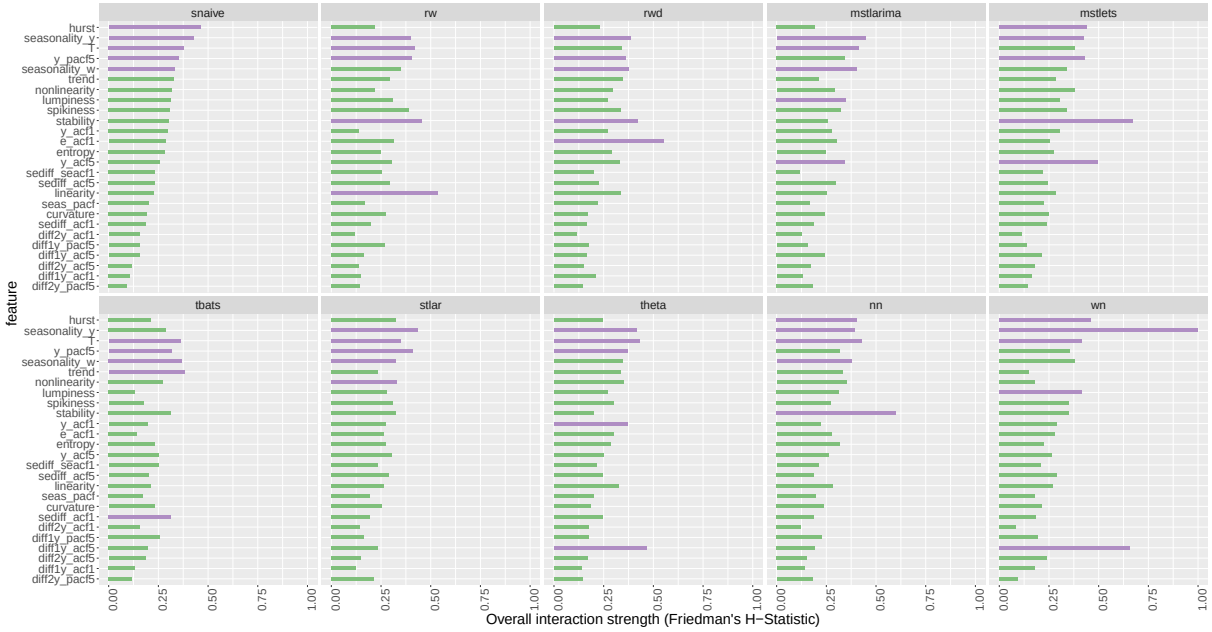

Figure 38: *The interaction strength (Friedman's H-Statistic) for each feature with any other feature for daily forest. Strength of annual seasonality shows highest level of interactivity within white noise class.*

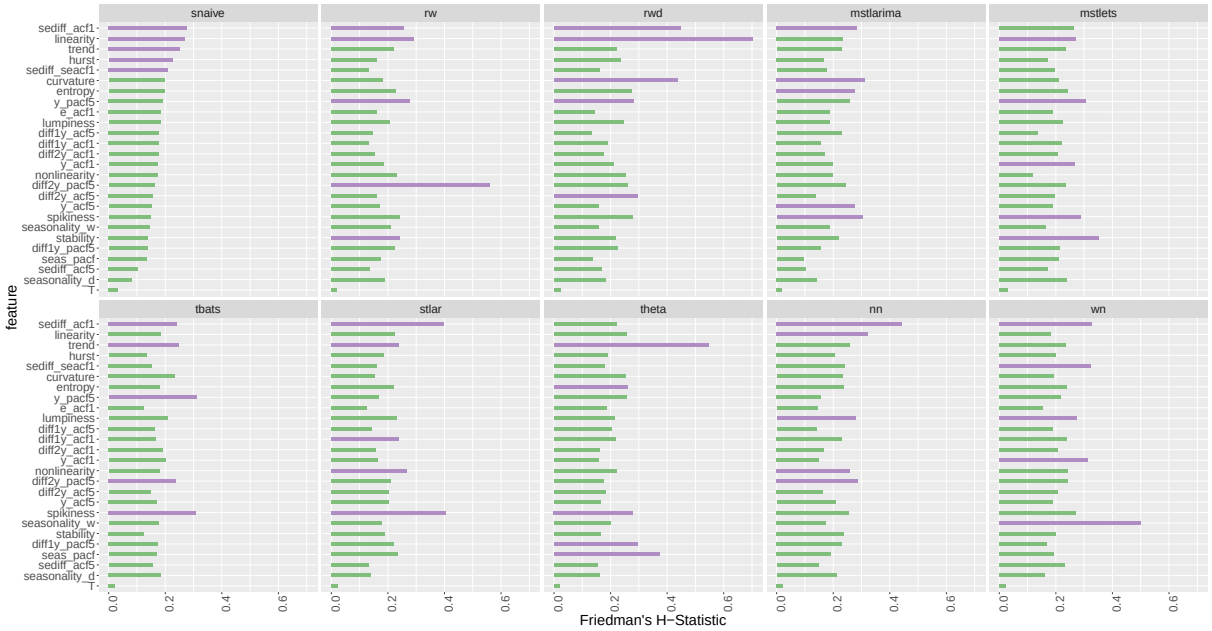

Figure 39: *The interaction strength (Friedman's H-Statistic) for each feature with any other feature for hourly forest. Linearity shows high level of interactivity within random walk with drift.*

Appendix B: Partial dependency plots for white noice class

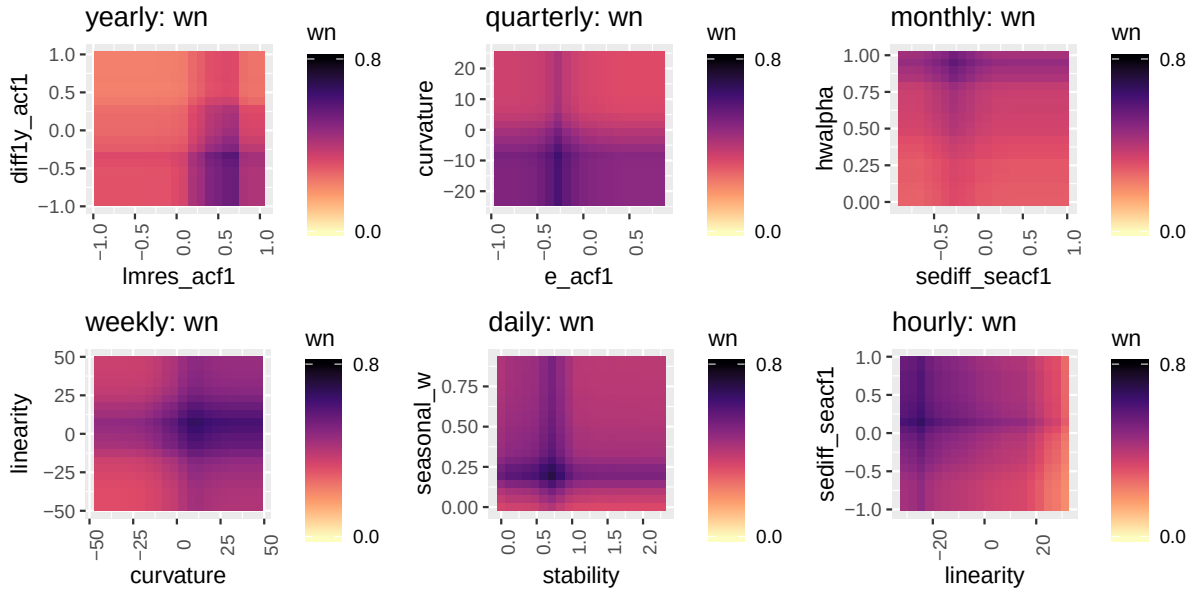

Figure 40: *Partial dependency plots for wn class*

Appendix C: Relative frequencies that each forecast model was selected as the best forecast model

Table 4: *Relative frequencies that each forecast model was selected as the best forecast model*

Class label	Description	Y	Q	M	W	D	H
wn	white noise process	1284	463	1403	3	156	1
ARMA	AR, MA, ARMA processes	97	24	11	-	0	-
ARIMA	ARIMA process	3351	351	649	-	2	-
SARIMA	seasonal ARIMA	-	3667	6782	54	-	-
rwd	random walk with drift	10754	9103	8955	41	671	0
rw	random walk	521	406	1731	10	323	0
theta	standard theta method	466	1071	3176	18	554	0
stlar		-	2369	9829	29	678	39
ETS_NTNS	ETS without trend and seasonal components	35	34	160	-	-	-
ETS_T	ETS with trend component and without seasonal component	4992	360	521	-	-	-
ETS_DT	ETS with damped trend component and without seasonal component	26	107	979	-	-	-
ETS_TS	ETS with trend and seasonal component	-	346	779	-	-	-
ETS_DTS	ETS with damped trend and seasonal component	-	47	640	-	-	-
ETS_S	ETS with seasonal component and without trend component	-	110	442	-	-	-
snaive	seasonal naive method	-	434	1247	20	93	50
tbats	TBATS forecasting	-	2880	5989	90	368	58
nn	neural network time series forecasts	1474	2128	4707	56	1010	119
mstlets		-	-	-	36	169	54
mstlarima		-	-	-	-	205	93

Appendix D: Summary of the results in Sections 7.1-7.6

Table 5: *Proportion of times each feature is selected for the top five ranks by different forecast model classes across each frequency category*

	Feature	Description	Y	Q	M	W	D	H
1	T	length of time series		0.05	0.52	0.33	0.70	0.10
2	trend	strength of trend	0.90	0.82	0.88	0.42	0.60	0.20
3	seasonality_q	strength of quarterly seasonality	-	1.00		-	-	-
4	seasonality_m	strength of monthly seasonality	-	-	1.00	-	-	-
5	seasonality_w	strength of weekly seasonality	-	-	-	0.66	0.50	0.40
6	seasonality_d	strength of daily seasonality	-	-		-		1.00
7	seasonality_y	strength of yearly seasonality	-	-		-	1.00	0.00
8	linearity	linearity	0.90	0.82	0.70	0.91	0.70	0.40
9	curvature	curvature	0.30	0.05	0.00	0.25	0.20	0.30
10	spikiness	spikiness	0.10		0.05	0.25	-	0.10
11	e_acf1	first ACF value of remainder series			0.05	0.25		0.10
12	stability	stability		0.11	0.47	0.41	0.80	0.20
13	lumpiness	lumpiness		0.11		0.08		
14	entropy	spectral entropy				0.25		0.60
15	hurst	Hurst exponent					0.10	
16	nonlinearity	nonlinearity				0.25		0.20
17	alpha	ETS(A,A,N) $\hat{\alpha}$		0.23	0.05		-	-
18	beta	ETS(A,A,N) $\hat{\beta}$	0.30	0.05	0.05	0.25	-	-
19	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-		0.11	-	-	
20	hwbeta	ETS(A,A,A) $\hat{\beta}$	-			-	-	
21	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	0.23		-	-	
22	ur_pp	test statistic based on Phillips-Perron test	0.90	-	-	-	-	-
23	ur_kpss	test statistic based on KPSS test		-	-	-	-	-
24	y_acf1	first ACF value of the original series	0.20					
25	diff1y_acf1	first ACF value of the differenced series	0.80	0.11	0.23		0.10	
26	diff2y_acf1	first ACF value of the twice-differenced series				0.08		0.20
27	y_acf5	sum of squares of first 5 ACF values of original series		0.05				
28	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	0.10	0.11	0.11		0.10	0.10
29	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series		0.05		0.08		0.10
30	sediff_acf1	ACF value at the first lag of seasonally-differenced series	-					0.10
31	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	0.11	0.17	0.25	0.10	0.20
32	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-		0.05		0.20	
33	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	-	0.05	0.05	0.25	
34	lmres_acf1	first ACF value of residual series of linear trend model	0.40	-	-	-	-	-
35	y_pacf5	sum of squares of first 5 PACF values of original series	0.40	0.52	0.29	0.33	0.90	0.90
36	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series		0.82			0.10	0.70
37	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series		0.05		0.08		

References

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

Collopy, F & JS Armstrong (1992). Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Friedman, JH, BE Popescu, et al. (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics* **2**(3), 916–954.

Goldstein, A, A Kapelner, J Bleich & E Pitkin (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics* **24**(1), 44–65.

Greenwell, BM, BC Boehmke & AJ McCarthy (2018). A Simple and effective model-based variable importance measure. *arXiv preprint arXiv: 1805.04755*.

Hyndman, R, G Athanasopoulos, C Bergmeir, G Caceres, L Chhay, M O'Hara-Wild, F Petropoulos, S Razbash, E Wang & F Yasmeen (2018). *forecast: Forecasting functions for time series and linear models*. R package version 8.5. http://pkg.robjhyndman.com/forecast.

Jiang, T & AB Owen (2002). Quasi-regression for visualization and interpretation of black box functions. *Technical Report, Stanford University.*

Kück, M, SF Crone & M Freitag (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp.1499–1506.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Lundberg, SM & SI Lee (2017). A unified approach to interpreting model predictions. In: pp.4765–4774.

M4 Competitor's Guide (2018). https://www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-Competitors-Guide.pdf. Accessed: 2018-09-26.

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

Meade, N (2000). Evidence for the selection of forecasting methods. *Journal of Forecasting* **19**(6), 515–535.

Petropoulos, F, S Makridakis, V Assimakopoulos & K Nikolopoulos (2014). 'Horses for courses' in demand forecasting. *European Journal of Operational Research* **237**(1), 152–163.

Prudêncio, RB & TB Ludermir (2004). Meta-learning approaches to selecting time series models. *Neurocomputing* **61**, 121–137.

Ribeiro, MT, S Singh & C Guestrin (2016). Why should I trust you?: Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY: ACM*, pp.1135–1144.

Schnaars, SP (1984). Situational factors affecting forecast accuracy. *Journal of Marketing Research*, 290–297.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Talagala, TS, RJ Hyndman & G Athanasopoulos (2018). Meta-learning how to forecast time series. *Technical Report 6/18, Monash University, Department of Econometrics and Business Statistics.*

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Zhao, Q & T Hastie (2017). Causal interpretations of black-box models. *Journal of Business & Economic Statistics*.

Chapter 4

FFORMA: Feature-based forecast model averaging

ARTICLE IN PRESS

International Journal of Forecasting xxx (xxxx) xxx

Contents lists available at ScienceDirect

International Journal of Forecasting

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

FFORMA: Feature-based forecast model averaging

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

Department of Econometrics and Business Statistics, Monash University, Australia

ARTICLE INFO

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

ABSTRACT

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

1. Introduction

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

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

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

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

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

2. Methodology

2.1. Intuition and overview of FFORMA

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

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

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

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

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

2.2. Algorithmic description

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

3. Implementation and application to the M4 competition

3.1. Reference set

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

3.2. Time series features

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

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

3.3. Pool of forecasting methods

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

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

Please cite this article as: P. Montero-Manso, G. Athanasopoulos, R.J. Hyndman et al., FFORMA: Feature-based forecast model averaging. International Journal of Forecasting (2019), https://doi.org/10.1016/j.ijforecast.2019.02.011.

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

OFFLINE PHASE: TRAIN THE LEARNING MODEL

Inputs:

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

Output:

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

Prepare the meta-data

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

Train the meta-learner, w

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

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

ONLINE PHASE: FORECAST A NEW TIME SERIES

Input:

FFORMA meta-learner from offline phase.

Output:

Forecast the new time series x_{new}.

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

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

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

3.4. Forecast loss measure

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

3.5. Meta-learning model implementation

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

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

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

Table 1
Features used in the FFORMA framework.

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

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

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

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

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

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

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

Please cite this article as: P. Montero-Manso, G. Athanasopoulos, R.J. Hyndman et al., FFORMA: Feature-based forecast model averaging. International Journal of Forecasting (2019), https://doi.org/10.1016/j.ijforecast.2019.02.011.

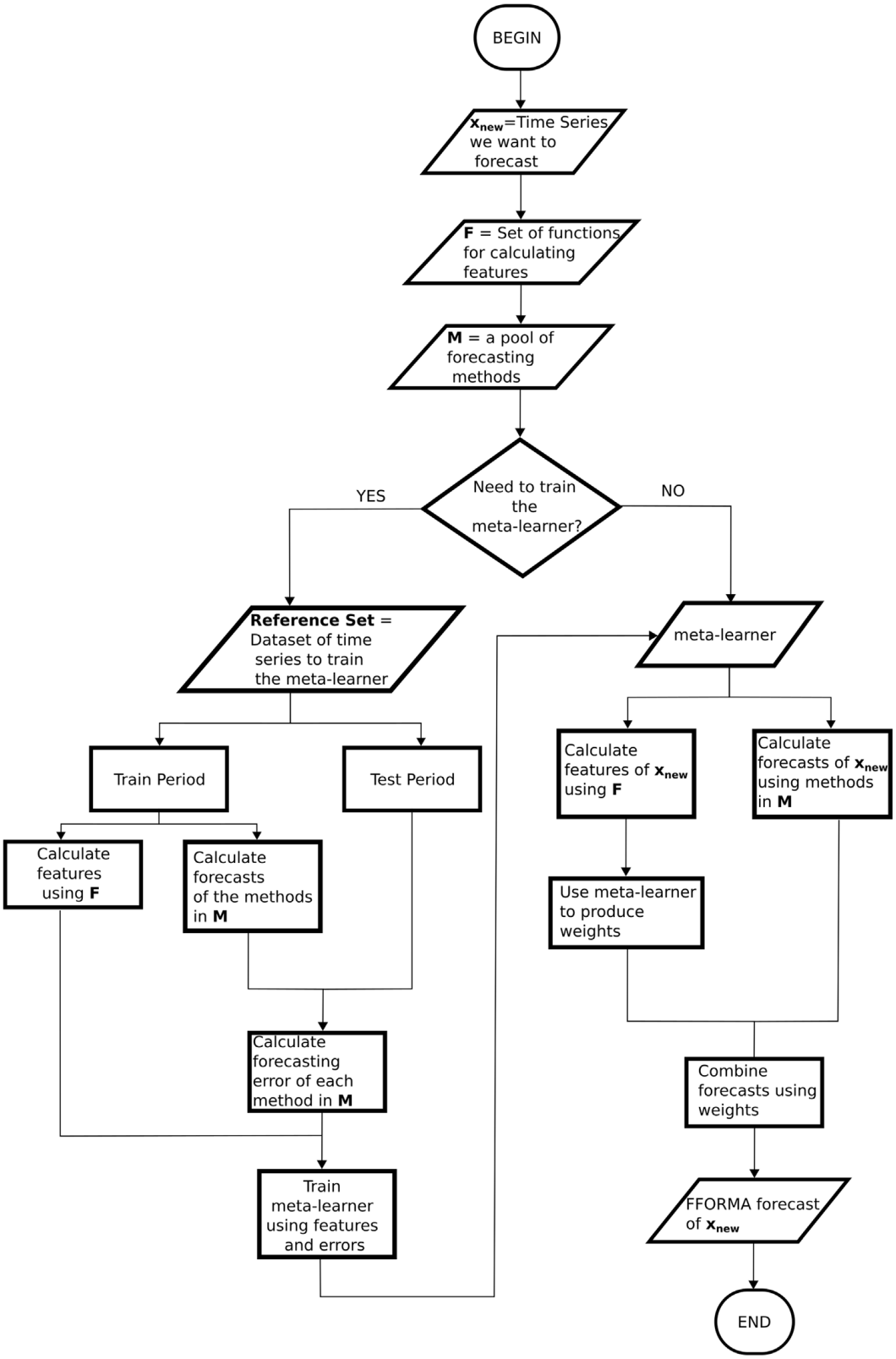

Fig. 1. Flowchart of the FFORMA forecasting process.

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

3.6. Prediction intervals

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

Please cite this article as: P. Montero-Manso, G. Athanasopoulos, R.J. Hyndman et al., FFORMA: Feature-based forecast model averaging. International Journal of Forecasting (2019), https://doi.org/10.1016/j.ijforecast.2019.02.011.

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

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

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

4. Discussion and conclusions

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

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

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

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

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

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

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

Acknowledgments

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

Please cite this article as: P. Montero-Manso, G. Athanasopoulos, R.J. Hyndman et al., FFORMA: Feature-based forecast model averaging. International Journal of Forecasting (2019), https://doi.org/10.1016/j.ijforecast.2019.02.011.

References

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

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). ACM.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Please cite this article as: P. Montero-Manso, G. Athanasopoulos, R.J. Hyndman et al., FFORMA: Feature-based forecast model averaging. International Journal of Forecasting (2019), https://doi.org/10.1016/j.ijforecast.2019.02.011.

Chapter 5

FFORMPP: Feature-based forecast model performance prediction

FFORMPP: Feature-based forecast model performance prediction

Thiyanga S. Talagala
Department of Econometrics and Business Statistics, Monash University, VIC 3800, Australia, and
ARC Centre of Excellence for Mathematics and Statistical Frontiers
Email: thiyanga.talagala@monash.edu
Corresponding author

Feng Li
School of Statistics and Mathematics, Central University of Finance and Economics, Beijing 100081, China
Email: feng.li@cufe.edu.cn

Yanfei Kang
School of Economics and Management, Beihang University, Beijing 100191, China
Email: yanfeikang@buaa.edu.cn

7 November 2019

JEL classification: C10,C14,C22

FFORMPP: Feature-based forecast model performance prediction

Abstract

This paper introduces a novel meta-learning algorithm for time series forecasting. The efficient Bayesian multivariate surface regression approach is used to model forecast error as a function of features calculated from the time series. The minimum predicted forecast error is then used to identify an individual model or combination of models to produce forecasts. In general, the performance of any meta-learner strongly depends on the reference dataset used to train the model. We further examine the feasibility of using GRATIS (a feature-based time series simulation approach) in generating a realistic time series collection to obtain a diverse collection of time series for our reference set. The proposed framework is tested using the M4 competition data and is compared against several benchmarks and other commonly used forecasting approaches. The new approach obtains performance comparable to the second and the third rankings of the M4 competition.

Keywords: Time series, meta-learning, mixture autoregressive models, surface regression, M4 Competition

1 Introduction

Forecasting is an important aspect of every business operation. The selection of a suitable forecast model or a combination of models to use in forecasting is at the heart of the forecasting process (Tashman & Leach 1991). This selection process is challenging in the context of large-scale time series forecasting for several reasons: i) there is no universal method that performs best for all kinds of forecasting problems; ii) a trial-and-error process of model selection would increase the time and computational cost significantly; iii) it is not possible to derive a typical algebraic expression for choosing the best-performing model(s) out of a portfolio of algorithms; and iv) even with expert knowledge, the correct solution is not guaranteed. A meta-learning approach serves as a promising alternative to solve this problem.

The idea of using meta-learning to select the best forecasting model for a given time series has been explored by several researchers in the context of time series forecasting (Collopy &

Armstrong 1992; Shah 1997; Adya et al. 2001; Wang, Smith-Miles & Hyndman 2009; Petropoulos et al. 2014). This approach is also known as an algorithm selection problem and can be expressed firmly using Rice's framework for algorithm selection (Rice 1976). Further evidence in favour of this idea is also given in Talagala, Hyndman & Athanasopoulos (2018). In all these cases, a vector of features computed from time series is used as input to train a meta-learner. The output and the objective function to train the meta-learner is approached differently. For example, Shah (1997) uses the best forecast model as the output label and applies a discriminant analysis to train a meta-learner to predict the forecast model that is expected to perform best on a given time series. The objective function is to minimise classification error. Further, Prudêncio & Ludermir (2004) use neural network approaches to define weights for the best linear combination of methods to improve forecast accuracy; thus, the objective function is to minimise forecast error.

Although many researchers have highlighted the usefulness of the meta-learning approach to guide the way the forecasts are computed, few studies have concluded that this approach is superior to simple benchmarks and commonly used forecasting approaches. For example, Meade (2000) concludes that the summary statistics are useful in selecting a good forecasting method, but are not necessarily the best. Two possible reasons for the infeasibility of the selection process are the use of inadequate features in the meta-learning process, and having training time series data that are not as diverse as required to predict different forecast model performance (Kang, Hyndman & Li 2019).

This paper is the third in a series of papers addressing the aforementioned issues in developing a meta-learning framework for forecast model selection based on features computed from the time series. Our first attempt to develop a framework for forecast model selection is described in Talagala, Hyndman & Athanasopoulos (2018). The first framework is called FFORMS: Feature-based FORecast Model-Selection. We use the random forest algorithm to predict the forecast model that is expected to perform best on a given time series. In our second paper, Montero-Manso et al. (2019), rather than mapping time series to a single forecast model we use a gradient boosting algorithm to obtain the weights for forecast combinations. We call our second framework FFORMA: Feature-based FORecast Model Averaging. FFORMA placed second in the M4 competition (Makridakis, Spiliotis & Assimakopoulos 2018). Having revisited the literature, we found that to the best of our knowledge none of these studies have considered the correlation structure of algorithm performance in their model training process. The current paper extends this idea. The third algorithm uses the efficient Bayesian multivariate surface regression approach to estimate forecast error for each method, and then uses the minimum predicted error to select a forecasting model or choose individual models for forecast combinations.

The main contributions of this paper are as follows:

1. We propose a novel meta-learning framework for time series forecasting. We refer to this general framework as FFORMPP: Feature-based FORecast Model Performance Prediction. It consists of two phases: the offline phase and the online phase. Most of the expensive computations for processing data and training a meta-learner are performed in the offline phase. We use a collection of time series to train a meta-learner. The data processing part requires computation of a set of features and forecast errors from a pool of forecast models for the time series in our collection. Subsequently, the efficient Bayesian multivariate surface regression approach proposed by Li & Villani (2013) is used to model forecast algorithm performances (measured by a forecast error measure, which in our case is MASE: Mean Absolute Scaled Error) as a function of features calculated from the time series. This produces a meta-learner to be used in the online phase. The online phase requires only the calculation of a simple vector of features for any newly given time series and uses the pre-trained classifier to estimate forecast error for each model in the pool, which is computationally efficient for real-time implementations. This allows ranking of the forecast models with respect to their forecast errors and evaluation of their relative forecast performance without calculating forecasts from all available individual models in the pool.

2. The diversity of features in the collection of time series used to train a meta-learner plays a critical role in training a meta-learning model. Often, time series with the required amount of feature diversity and quality might not be available (Kang, Hyndman & Smith-Miles 2017; Kang, Hyndman & Li 2019). To this end, we explore the use of GRATIS (GeneRAting TIme Series with diverse and controllable characteristics) proposed by Kang, Hyndman & Li (2019) to obtain a diverse collection of time series.

3. We visualise time series in the instance space defined by the features according to the forecast models used to compute combination forecasts. This helps us to explore the distribution of locations of the time series in the instance space and their relationship with features and forecast model selection.

The remainder of the paper is organised as follows: Section 2 introduces the methodology, including the methodology for simulating time series to augment the reference set to train a model and efficient Bayesian multivariate regression approach. Section 3 discusses the results in application to the M4 competition data. Section 4 concludes the paper.

2 Methodology

Figure 1 shows the proposed framework. For each series, a set of features computed from the training period of each series comprising an input vector and the MASE values for each method gives the output vector to train a model. The description of the features calculated in each frequency category is shown in Table 1. We analyse yearly, quarterly, monthly, weekly, daily and hourly series separately. Table 2 shows the forecast models we consider within each frequency category.

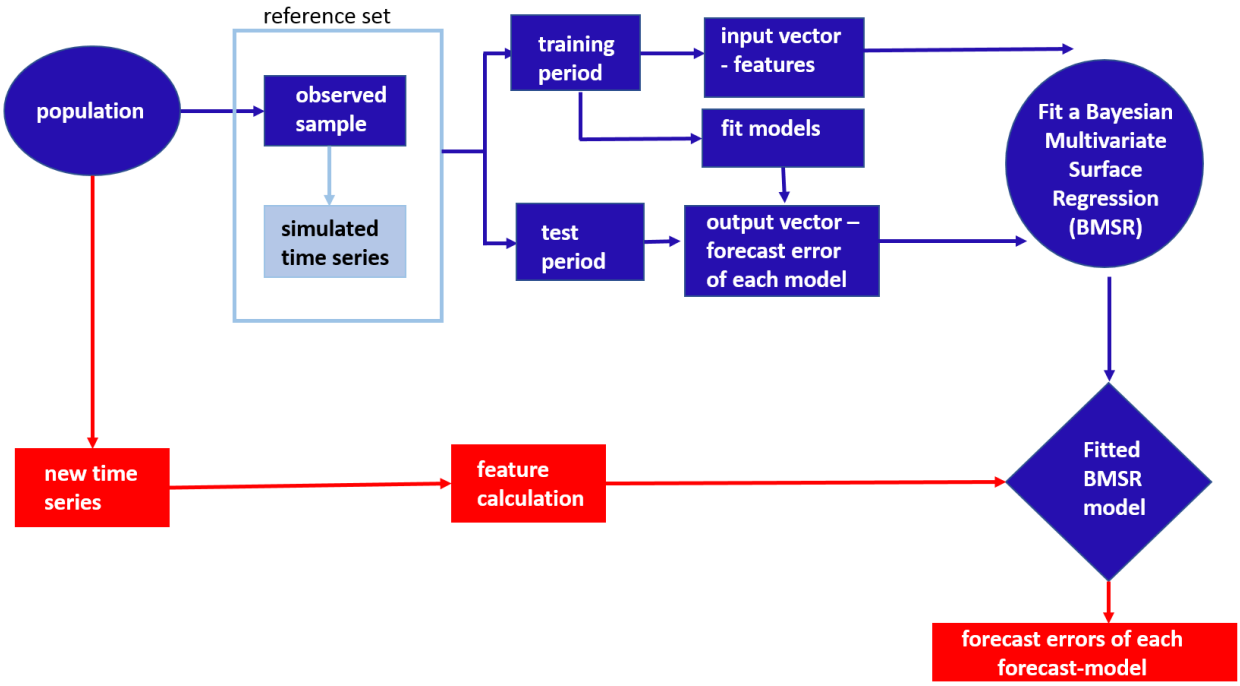

Figure 1: *FFORMPP (Feature-based FORecast Model Performance Prediction) framework. The offline phase is shown in blue and the online phase in red.*

2.1 Reference set

We use the time series of M1 and M3 competitions as the observed sample for yearly, quarterly and monthly time series. In addition to the observed time series, we simulate 10, 000 time series based on MAR (Mixture AutoRegressive) models introduced by Kang, Hyndman & Li (2019). The observed time series and the simulated time series form the reference set to fit the model. For weekly, daily and hourly frequencies, only the simulated time series are used to create the reference set because the previous M-competitions do not contain the time series corresponding to weekly, daily and hourly frequencies. Once we have formed the reference set, we split each time series in the reference set into training period and test period. Features are calculated based on the training period of each series and the forecast error measure is calculated over the test period of each time series. Table 3 summarises the number of time series in the reference set and

Table 1: *Time series features*

	Feature	Description	Y	Q/M	W	D/H
1	T	length of time series	✓	✓	✓	✓
2	trend	strength of trend	✓	✓	✓	✓
3	seasonality_q	strength of quarterly seasonality	-	✓	-	-
4	seasonality_m	strength of monthly seasonality	-	✓	-	-
5	seasonality_w	strength of weekly seasonality	-	-	✓	✓
6	seasonality_d	strength of daily seasonality	-	-	-	✓
7	seasonality_y	strength of yearly seasonality	-	-	-	✓
8	linearity	linearity	✓	✓	✓	✓
9	curvature	curvature	✓	✓	✓	✓
10	spikiness	spikiness	✓	✓	✓	✓
11	e_acf1	first ACF value of remainder series	✓	✓	✓	✓
12	stability	stability	✓	✓	✓	✓
13	lumpiness	lumpiness	✓	✓	✓	✓
14	entropy	spectral entropy	✓	✓	✓	✓
15	hurst	Hurst exponent	✓	✓	✓	✓
16	nonlinearity	nonlinearity	✓	✓	✓	✓
17	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓	✓	-
18	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓	✓	-
19	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓	-	-
20	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓	-	-
21	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓	-	-
22	ur_pp	test statistic based on Phillips-Perron test	✓	-	-	-
23	ur_kpss	test statistic based on KPSS test	✓	-	-	-
24	y_acf1	first ACF value of the original series	✓	✓	✓	✓
25	diff1y_acf1	first ACF value of the differenced series	✓	✓	✓	✓
26	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓	✓	✓
27	y_acf5	sum of squares of first 5 ACF values of original series	✓	✓	✓	✓
28	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓	✓	✓
29	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓	✓	✓
30	sediff_acf1	ACF value at the first lag of seasonally-differenced series	-	✓	✓	✓
31	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓	✓	✓
32	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓	✓	✓
33	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	✓	✓	✓
34	lmres_acf1	first ACF value of residual series of linear trend model	✓	-	-	-
35	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓	✓	✓
36	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓	✓	✓
37	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓	✓	✓

Table 2: *Class labels*

class label	Description	Y	Q/M	W	D/H
WN	white noise process	✓	✓	✓	✓
auto.arima	ARIMA processes	✓	✓	✓	-
ets	exponential smoothing	✓	✓	-	-
rwd	random walk with drift	✓	✓	✓	✓
rw	random walk	✓	✓	✓	✓
theta	standard theta method	✓	✓	✓	✓
stlar	STL-AR method	-	✓	✓	✓
snaive	seasonal naive method	-	✓	✓	✓
tbats	TBATS forecasting	-	✓	✓	✓
nn	neural network time series forecasts	✓	✓	✓	✓
mstlets	multiple seasonal time series	-	-	✓	✓
mstlarima	multiple seasonal time series	-	-	-	✓

the new time series in each frequency category. Note that for yearly, quarterly and monthly time series, the reference set used to train a meta-learner is much smaller than the test set evaluating the meta-learner. This is often the case when applying the meta-learner in practice during the online phase.

Table 3: *Composition of the time series in the reference set and collection of new time series*

Frequency	Reference set			New series
	M1	M3	Simulated	M4
Yearly	181	645	10000	23000
Quarterly	203	756	10000	24000
Monthly	617	1428	10000	48000
Weekly	-	-	10000	359
Daily	-	-	10000	4227
Hourly	-	-	10000	414

2.2 Augmenting the observed sample with simulated time series from mixture autoregressive models

The reference set is augmented with simulated time series to obtain a more heterogeneous collection of time series for training a model. This helps to reduce overfitting to a relatively homogeneous set of data and increases generalisability of the model when applied to new time series with different conditions. Further, this is useful when there is no observed set of time series available to train a model in the offline phase. In this study, the simulated time series are generated based on the algorithm proposed by Kang, Hyndman & Li (2019), hereafter referred to as GRATIS[1]. Although there is no standard process for simulating time series, the most common approach involves simulation based on some data-generating processes (DGPs) such as exponential smoothing and ARIMA models. Instead of relying on a set of DGPs to generate time series, the GRATIS algorithm simulates time series based on a diverse set of time series features using MAR models. The algorithm can also allow a user to set controllable features for simulating time series. Table 4 lists the choice of values for parameters used to simulate time series in each frequency category. We used frequencies 1, 4, 12 and 52 to generate yearly, quarterly, monthly and weekly series respectively. Daily and hourly time series with a long history often show multiple seasonal patterns. Hence, for daily series, frequencies were set to 7 (time-of-week pattern) and 365.25 (annual seasonality), while for hourly series frequencies were set to 24 (time-of-day pattern) and 168 (time-of-week pattern). None of the time series in our hourly test dataset are longer than 8760. Hence, time-of-year pattern ($365 \times 24 = 8760$) was not considered. Except for hourly series, length of the time series is randomly chosen from

[1]The R package `tsgeneration` accompanies this work and is publicly available on https://github.com/ykang/tsgeneration.

uniform distribution. The minimum and maximum values of the distributions are selected based on lower (Q1 - 1.5 $\times$ IQR) and upper (Q3 + 1.5 $\times$ IQR) edges of the box-and-whisker plots. We use the M4 competition data to compute the associated statistics for length. The corresponding distributions are shown in Figure 2. The reason for this choice is that we use length of the time series as a feature in our meta-learning framework. Hence, lengths of the series in our reference set should cover all or the majority of time series we need to forecast. In practice, it might be difficult to obtain a reference set covering the whole length of the new time series for two main reasons: i) information regarding the range of length is not available at the offline stage (although, a rough idea about the distribution of the majority can be obtained); and ii) the range of length is very wide owing to 'outlying' observations (for example, Figure 2, quarterly and monthly series). In such circumstances, a reference set covering most of the lengths of future time series is a reasonable approach. Further, for each series we randomly select a number of mixing components from $\{1, 2, 3, 4, 5\}$. Other parameters are set the same as those in Kang, Hyndman & Li (2019). Having each parameter set as in Table 4, we generate 10000 time series from each frequency category. This is to reduce the training time of the efficient Bayesian multivariate surface regression model while maintaining the diversity of the reference set.

Table 4: *GRATIS: Choice of values for parameters for simulating time series using MAR models*

Value	Parameter	Description
frequency	seasonal period	yearly 1 quarterly 4 monthly 12 weekly 52 daily (7, 365) hourly (24, 168)
length	length of time series	yearly U(19, 75) quarterly U(24, 202) monthly U(60, 660) weekly U(93, 2610) daily U(107, 9933) hourly: 40.8% with 748 length and 59.2% with 1008 length
k	number of mixing components	For each series randomly chosen from $\{1, 2, 3, 4, 5\}$
α_k	weights of mixture components	$\alpha_k = \beta_k / \sum_{i=1}^{K} \beta_i$, where $\beta_i \sim U(0, 1)$
θ_{ki}	coefficients of the AR part	N(0, 0.5)
Θ_{kj}	coefficients of the seasonal AR parts	N(0, 0.5)
d_k	number of differences in each component	Bernoulli(0.9)
D_k	number of seasonal differences in each component	Bernoulli(0.4)

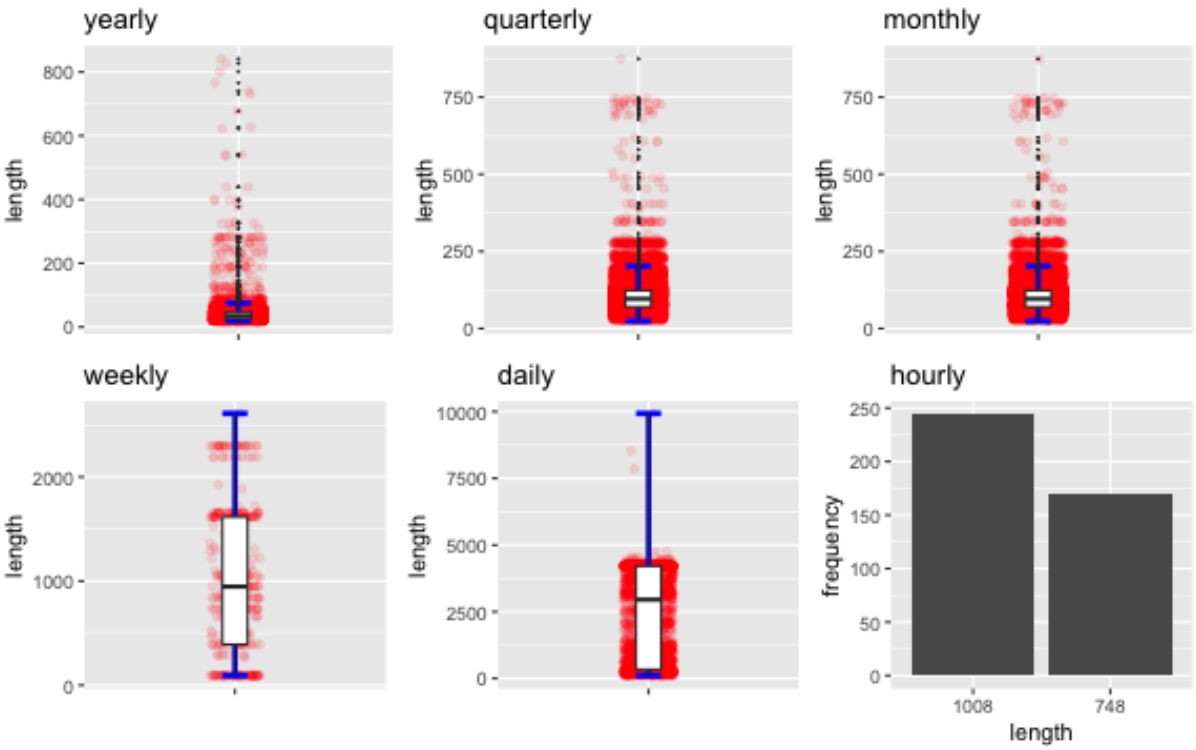

Figure 2: *Distribution of length of time series of the M4 competition across different frequency categories. The data points are shown in red. Horizontal blue solid line shows the lower and the upper whisker borders of each boxplot. Some outliers could be observed in yearly, quarterly and monthly categories. There are only two distinct levels of length for hourly series.*

2.3 Efficient Bayesian multivariate surface regression approach

The most commonly used additive spline regression assumes additivity in the regressor; that is, $E(y|x_1,...,x_q) = \sum_{j=1}^{q} f_j(x_j)$, where $f_j(x_j)$ is a spline regressor of the j^{th} regressor. Even though the assumption of additivity simplifies the model, it is quite a restrictive assumption. This problem has motivated research on surface models with interactions between regressors (Li & Villani 2013). Li & Villani (2013) proposed a general Bayesian approach for fitting surface models for a continuous multivariate response by combining additive splines and interactive splines. The proposed modelling is called efficient Bayesian multivariate surface regression.

The number of knots has an important influence in the resulting fit of spline regression: without enough knots the regression is underfitted and with too many knots it is overfitted. Choosing the locations of knots is also a challenge. This becomes even harder for surface regression than it is for additive models because any feasible set of q-dimensional knots is necessarily sparse in R^q when the number of regressors, q, is moderate or large. This causes the curse of dimensionality. The most common approach used in the literature is to use a fixed set of knot locations, and most of these algorithms place the knots at the centroids of the clusters computed based on regressor observations. Li & Villani (2013) pointed out that this is impractical when estimating a surface

with several regressors. Hence, the authors proposed a computationally efficient Markov chain Monte Carlo (MCMC) algorithm for the Gaussian multivariate surface regression to update the locations of the knots jointly. Instead of a fixed set of locations, the authors introduce 'moving knots'.

The proposed Gaussian multivariate regression model can be written as follows:

$$\boldsymbol{Y} = \boldsymbol{X_0 B_0} + \boldsymbol{X}_a(\boldsymbol{\xi}_a)\boldsymbol{B}_a + \boldsymbol{X}_s(\boldsymbol{\xi}_s)\boldsymbol{B}_s + \boldsymbol{E}, \tag{1}$$

where $\mathbf{Y}$ is a matrix of n number of observations and p number of response variables. The rows of $\mathbf{E}$ are error vectors assumed to be independent and identically distributed (iid) as $N_p(0, \Sigma)$. The proposed efficient Bayesian multivariate surface regression model contains three components:

1. **Linear component:** The linear component contains the original covariates including the constant term. This enters the model in linear form. The matrix $\mathbf{X_0}$ is a $n \times q_0$ vector in which the first column contains ones for the intercept. The corresponding regression coefficients are in $\mathbf{B_o}$.

2. **Additive component:** The second component of the model contains additive spline basis functions of the covariates in $\mathbf{X_0}$. This is represented by $\boldsymbol{X}_a(\boldsymbol{\xi}_a)$, where ξ_a represents the knots. It is important to note that the knots in the additive part of the model are scalars and this model allows an unequal number of knots for different covariates in the model. The matrix $\mathbf{B_a}$ contains the regression coefficients corresponding to the additive component. The additive component of the model captures the non-linear relationship between features and response $\mathbf{Y}$.

3. **Surface component:** The surface component of the model contains the radial basis function for capturing the remaining part of the surface and interactions. This is denoted by $\boldsymbol{X}_s(\boldsymbol{\xi}_s)$. Note that the ξ_s is a q_0-dimensional vector. The matrix $\mathbf{B_s}$ contains the regression coefficients corresponding to the surface component and ξ_s represents the surface knots.

For notational convenience Equation (1) can be written as

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}, \tag{2}$$

where $\mathbf{X} = [\mathbf{X_0}, \mathbf{X_a}, \mathbf{X_s}]$ is the $n \times q$ design matrix ($q = q_0 + q_a + q_s$) and $\mathbf{B} = [\mathbf{B_0'}, \mathbf{B_a'}, \mathbf{B_s'}]$. For a given set of fixed knot locations, the model in Equation (1) is linear in regression parameters. The

over-parameterised problem is addressed by using shrinkage priors to shrink small regression coefficients towards zero. The shrinkage parameters and knot locations of ξ_a and ξ_s are treated as unknown parameters to be estimated. Li & Villani (2013) proposed a computationally efficient MCMC algorithm to estimate shrinkage parameters and update the locations of the knots of additive and surface components jointly. This approach allows an unequal number of knots in the different covariates. In addition, separate shrinkage parameters for the linear, additive and surface parts of the model are allowed. Further, this approach permitted separate shrinkage parameters for the p responses within each of the three model parts. In contrast to other spline-bases models, this approach allows the knot locations to move freely in the regressor space, and thus fewer knots are usually required. The estimation and computation details can be found in Li & Villani (2013).

3 Application to the M4 competition data

3.1 Dissimilarity between different datasets

Principal component analysis preserves the dissimilarity between widely separated data points rather than the similarity between nearby data points. This feature is useful for improved confidence in simulated series' representativeness of real time series. For example, if the simulated time series results in isolated clusters or highly dense clusters far apart from the real-world time series, it indicates a poor representation of the real data.

We use principal component analysis to visualise dissimilarity between the datasets: observed time series (M1 and M3), simulated series and new time series (M4) in the two-dimensional instance space. The data were normalized using the z-scale transformation before applying PCA. This also helps us to gain an idea about the global structure of the location of the different collections. The results for yearly, quarterly and monthly series are shown in Figure 3. We compute principal components using the time series in the reference set (observed series and simulated series) and project new series (M4) into the two-dimensional space spanned by the first two eigen vectors. The first two principal components explain 51.3%, 48.7% and 48.3% of the total variation in the yearly, quarterly and monthly data. We see that the distribution of the simulated time series (represented by the dark orange dots) clearly nests and fills in the instance space. The simulated time series fills the instance space by further expanding the density range of observed time series rather than resulting in isolated clusters. This guarantees that the simulated data generated based on the GRATIS approach are actually representative of real data. Further, we can see that the projection of M4 series falls within the space created

by the series in the reference set (M1, M3 and simulated). This is very important because our FFORMS framework is trained based on the series in the reference set; hence, the model is valid over the space of the reference set. A few M4 time series in the monthly frequency category fall outside the convex hull of the reference set in the first two principal components owing very high length. Note that for yearly, quarterly and monthly series the size of the M4 time series collection is much greater than the size of the corresponding collection of simulated series. For yearly and quarterly data, the number of time series in the M4 collection is about twice as large as the simulated series, and for monthly data the M4 competition collection is four times larger than the simulated series. This shows the efficiency of the GRATIS simulation approach in increasing the diversity of feature space without having many time series similar in size to the new time series collection from which we wish to produce forecasts.

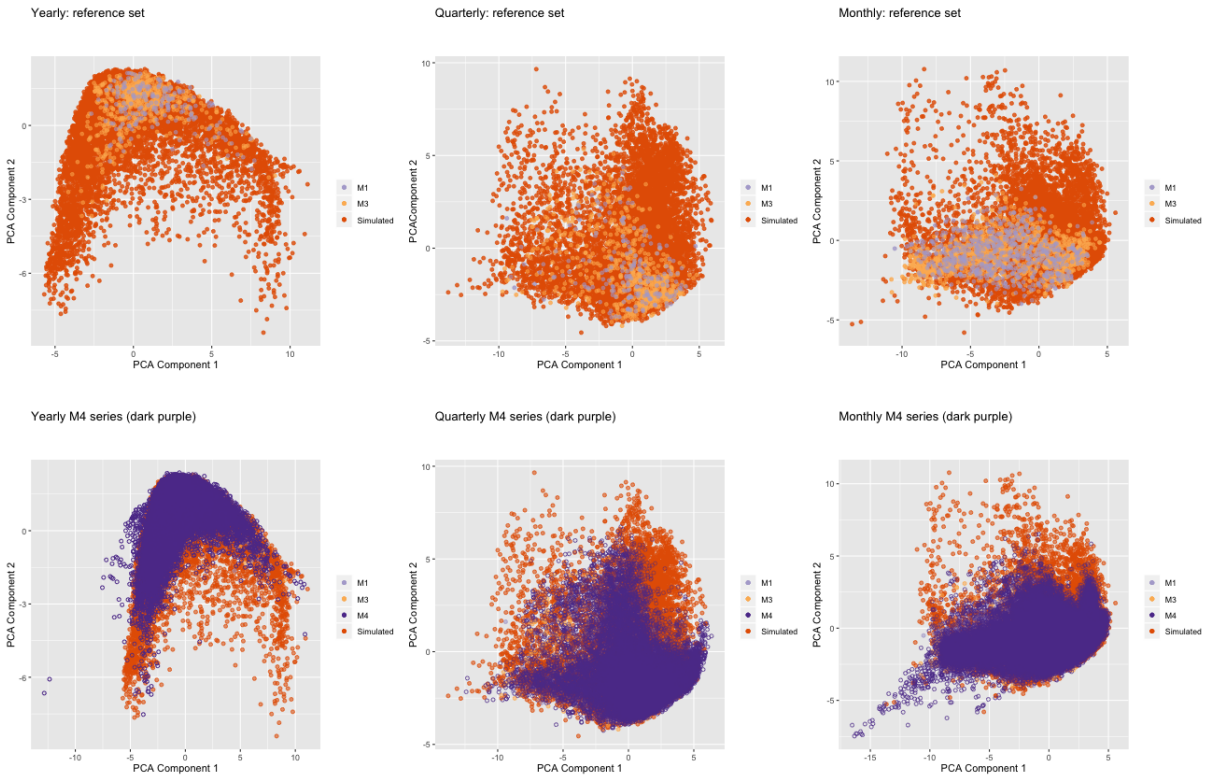

Figure 3: *Distribution of yearly, quarterly and monthly time series on the PCA space. On each graph, dark orange represents the simulated series and purple denotes new time series (M4 data). In the first row observed time series in the reference are highlighted in light orange and light purple. PCA space is computed based on the time series in the reference set and then the M4 competition (new series) are projected into the two-dimensional PCA space. Except for a few series, the majority of new time series we need to forecast fall within the space of the reference set. Note that for yearly, quarterly and monthly series the size of the test set is much larger than the size of the corresponding reference set.*

Figure 4 shows the principal component projections of weekly, daily and hourly time series. The first two principal components explain 48.7%, 43.4% and 45.4% of the total variance of

weekly, daily and hourly series. In all three frequency categories, M4-competition data fall within the feature space of simulated series. For weekly, daily and hourly frequency categories, simulated time series based on the GRATIS approach successfully fill in the space of the new time series. This shows the efficiency of the GRATIS approach in generating a realistic set of representative time series. This also shows that the GRATIS approach is a good choice to augment the reference set when only small amounts of training data are available. However, daily series of M4 competition series are clustered into a single location of the feature space. This is due to the quality issues of the M4 competition data. Daily series in the M4 competition are very similar to each other owing to data leakages. Examples of data leakage include, use of different segments of time series to create a new time series, and addition of a constant value to create new time series (Ingel et al. 2019). This reveals that the GRATIS simulated-based approach can also be used to evaluate the quality issues in the data.

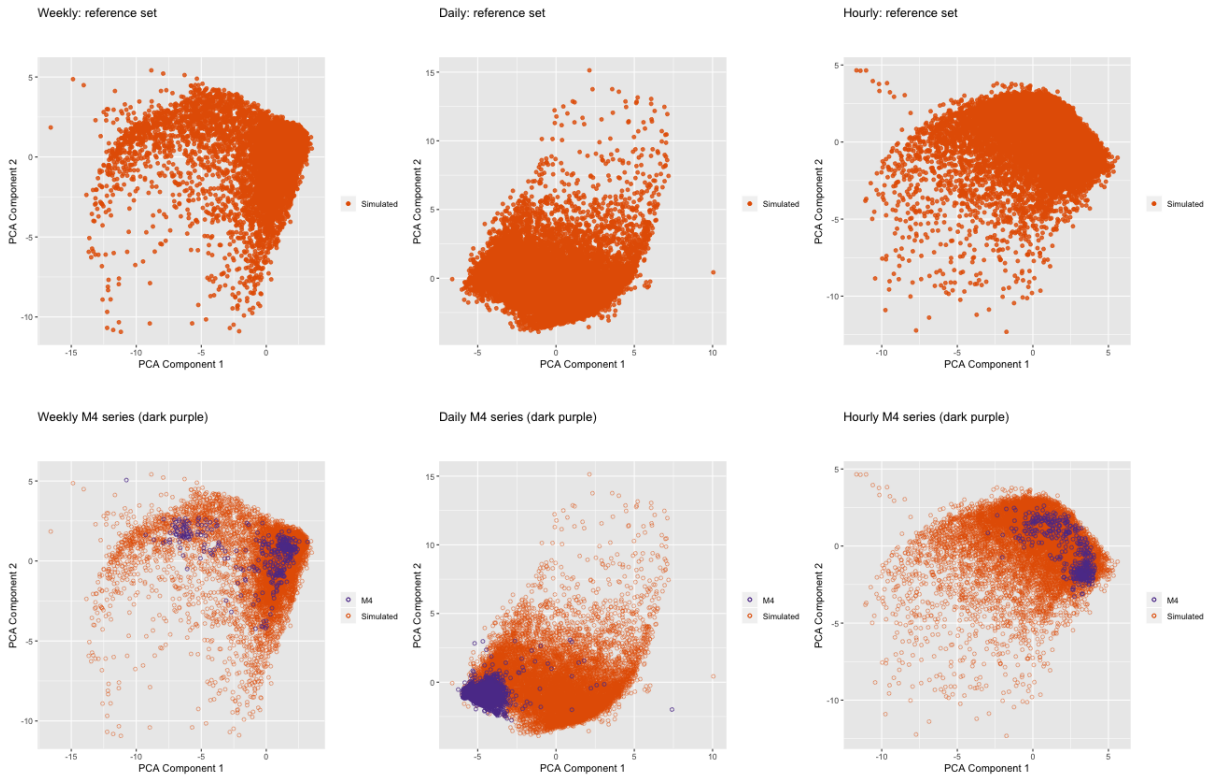

Figure 4: *Distribution of weekly, daily and hourly time series on the PCA space. On each graph, dark orange represents the simulated series and purple denotes new time series (M4 data). PCA space is computed based on the time series in the reference set and then the M4 competition (new series) are projected into the two-dimensional PCA space. In each frequency category, new series fall within the PCA space of reference set.*

3.2 Coverage analysis

To obtain a more accurate estimate of the exact coverage of simulated data, we adapt the idea used by Kang, Hyndman & Li (2019). For this purpose, the t-Stochastic Neighbor Embedding (t-SNE) approach is adapted. The t-SNE is a non-linear dimension reduction technique that seeks to find a low-dimensional sub-manifold that preserves the local structure of the dataset. In other words, the t-SNE approach keeps the neighbouring points in the original data space close in the embedding space. To quantify the miscoverage, first a grid of 900 squares was superimposed on the region of space covered by each scatter plot shown in Figure 5. Then the miscoverage of dataset A over dataset B is computed as follows (Kang, Hyndman & Li 2019):

$$miscoverage_{A/B} = N^{-2} \sum_{i=1}^{N^2} (1 - I_{i,A}) \times I_{i,B},$$

where $I_{i,A} = 1$ if points in dataset A fall within i^{th} square and $I_{i,A} = 0$ otherwise. An analogous definition is applied to $I_{i,B}$ computed based on the dataset B. We computed miscoverage of the simulated dataset over the M4 dataset and vice versa. The results are shown in Table 5. Table 5 shows that our attempt to increase the diversity of the reference set using the GRATIS approach is successful.

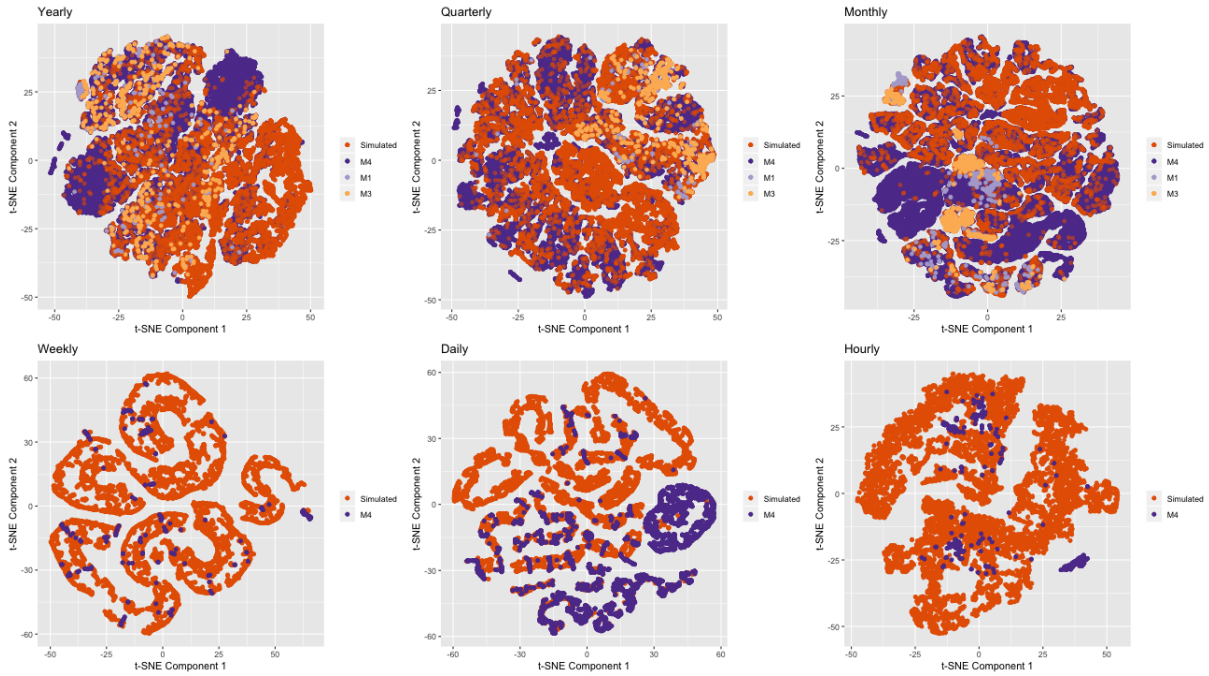

Figure 5: *Two-dimensional t-SNE instance spaces of the M1, M3, M4 and simulated series for yearly, quarterly, monthly, weekly, daily and hourly series.*

Table 5: *Coverage analysis*

	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
Miscoverage of the 'Simulated' dataset over the 'M4' dataset	0.068	0.043	0.132	0.001	0.061	0.011
Miscoverage of the 'M4' dataset over the 'Simulated' dataset	0.026	0.022	0.011	0.438	0.278	0.490

3.3 Forecasting results

We evaluate the out-of-sample performances of our proposed meta-learning framework to the benchmark based on the M4 competition series. The mean absolute scale error (MASE) (Hyndman & Koehler 2006) was used to evaluate forecast accuracy. We compute both individual and combination forecasts for the M4 competition series. Individual forecasts are computed based on the model corresponding to the minimum predicted MASE. The combination forecasts are computed by taking the median of individual forecasts corresponding to the four models with minimum predicted MASE. The results are shown in Table 6. The results suggest that the forecast combination obtained through the FFORMPP framework outperforms the corresponding benchmark and other commonly used methods of forecasting. Further, we compare our results with those of the top three places of the M4 competition. The winning method of the competition is based on the Hybrid approach, which is a combination of exponential smoothing models and neural network approaches (Makridakis, Spiliotis & Assimakopoulos 2018). The second and third approaches are based on combination forecasts computed based on nine and seven individual forecast models respectively. According to the results of Table 6, we can see that our approach achieved comparable results in a much more cost- and time-effective manner because our combination forecasts are calculated based on four individual models.

We also reported the relative frequencies with which each forecast model is selected as a component of the calculation of combination forecasts based on FFORMPP. Further, to gain an idea of the different types of model combinations used to compute forecast, we cluster time series based on the models that are used to compute FFORMPP combination. For this purpose, we first create a design matrix of 1 and 0. The columns of the design matrix correspond to each individual forecast model, and rows correspond to each time series. The cell values of the matrix are 1s and 0s, where 1 is assigned if a corresponding model is used for combination forecast, and 0 otherwise. Then, hierarchical clustering was done by using a binary distance metric and ward clustering method. A cluster analysis was performed separately for each frequency category.

For each frequency category we identified three main clusters. The results are shown in Table 7

Table 6: *MASE values calculated over the M4 competition data*

	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMPP-combination*	**3.07**	**1.13**	**0.89**	**2.46**	3.62	**0.96**
FFORMPP-individual	3.37	1.17	1.05	2.53	4.26	1.06
auto.arima	3.40	1.17	0.93	2.55	-	-
ets	3.44	1.16	0.95	-	-	-
theta	3.37	1.24	0.97	2.64	3.33	1.59
rwd	**3.07**	1.33	1.18	2.68	3.25	11.45
rw	3.97	1.48	1.21	2.78	**3.27**	11.60
nn	4.06	1.55	1.14	4.04	3.90	1.09
stlar	-	2.02	1.33	3.15	4.49	1.49
snaive	-	1.66	1.26	2.78	24.46	2.86
tbats	-	1.19	1.05	2.49	**3.27**	1.30
wn	13.42	6.50	4.11	49.91	38.07	11.68
mstlarima	-	-	-	-	3.84	1.12
mstlets	-	-	-	-	3.73	1.23
combination (median)	3.29	1.22	0.95	2.57	3.52	1.33
combination (mean)	4.09	1.58	1.16	6.96	7.94	3.93
M4 competition top 3 places (MASE)						
M4-1st	2.98	1.12	0.88	2.36	3.45	0.89
M4-2nd	3.06	1.11	0.89	2.11	3.34	0.81
M4-3rd	3.13	1.12	0.91	2.16	2.64	0.87

and the associated graphical representation is shown in Figure 6. According to the results of Table 6, for yearly data we can see that `auto.arima`, `ets`, `theta` and `rwd` give the best individual forecast. From Table 7 we can see that those four models were most frequently selected to the FFORMPP combination forecast. Similarly, from Table 6 we can see that for quarterly and monthly series `auto.arima`, `ets` and `tbats` provide the best individual forecasts, and according to Table 7, we can see that those models are selected most often (approximately greater than 75%) for quarterly and monthly series. Similarly, we can interpret the results for weekly, daily and hourly series. Cluster 3 in the yearly series is very similar to cluster 2, the only difference being that, cluster 3 uses `nn` instead of `theta` model. The series in the first cluster uses a different combination of models from that which we considered for the yearly series, apart from the two combinations used in cluster 2 and cluster 3 respectively. For quarterly series, the biggest cluster is cluster 3, in which `auto.arima`, `ets`, `rwd` and `tbats` are used to calculate combination forecasts. Similar to the results of quarterly series, for monthly and daily series we observe two clusters that are homogeneous in terms of models used to compute combination forecasts. In terms of weekly data, most of the series use `auto.arima`, `theta`, `rwd` and `tbats` for combination forecasts. For daily and hourly series, it is interesting to observe that all the series in cluster 2 and cluster 3 use at least one of the models `mstlarima` or `mstlets`, which handle multiple seasonality, as a component in calculating combination forecast.

Table 7: *Relative frequencies that each forecast model was selected as a component to the calculation of combination forecasts based on FFORMPP (All values are shown in percentages).*

Source	No. of series	auto.arima	ets	theta	rwd	rw	nn	stlar	snaive	tbats	wn	mstlarima	mstlets
Yearly	23000	**82.7**	**80.4**	**79.2**	**79.1**	33.7	34.6	-	-	-	10.2	-	-
Cluster 1	9287	57.2	51.5	84.3	48.3	83.5	49.9	-	-	-	25.3	-	-
Cluster 2	10391	100	100	100	100	0	0	-	-	-	0	-	-
Cluster 3	3322	100	100	0	100	0	100	-	-	-	0	-	-
Quarterly	24000	**93.1**	**82.8**	13.8	47.8	11.4	35.8	12.9	9.1	**87.4**	5.8	-	-
Cluster 1	7126	76.6	42.1	46.6	22.5	38.4	22.6	43.7	30.5	57.6	19.5	-	-
Cluster 2	6994	100	100	0	0	0	100	0	0	100	0	-	-
Cluster 3	9880	100	100	0	100	0	0	0	0	100	0	-	-
Monthly*	48000	**88.9**	**74.4**	30.6	45.4	16.1	31.8	19.9	7.1	**83.1**	2.8	-	-
Cluster 1	12615	74.1	40.0	72.0	20.9	37.5	25.5	46.8	16.5	60.1	6.5	-	-
Cluster 2	11084	100	100	0	100	0	0	0	0	100	0	-	-
Cluster 3	6301	100	100	0	0	0	100	0	0	100	0	-	-
Weekly	359	**94.4**	-	**74.7**	69.4	9.8	31.5	29.5	10.9	**75.5**	4.5	-	-
Cluster 1	105	80.9	-	57.1	6.7	76.2	63.8	100	37.1	31.4	15.2	-	-
Cluster 2	68	100	-	32.4	82.4	39.7	67.6	1.5	0	76.5	0	-	-
Cluster 3	186	100	-	100	100	0	0	0	0	100	0	-	-
Daily	4227	-	-	18.4	8.7	9.5	**85.9**	**72.9**	5.63	**87.7**	4.1	65.8	44.3
Cluster 1	1891	-	-	34.5	19.5	21.2	68.5	39.9	12.6	72.6	9.1	65.4	52.3
Cluster 2	791	-	-	0	0	0	100	100	0	100	0	0	100
Cluster 3	1545	-	-	0	0	0	100	100	0	100	0	100	0
Hourly	414	-	-	2.9	4.8	16.2	68.8	57.7	14.3	**91.3**	21.0	**78.9**	43.9
Cluster 1	151	-	-	40.4	29.1	25.2	80.1	50.3	19.2	76.2	21.2	46.4	11.9
Cluster 2	162	-	-	0	0	0	100	100	0	0	0	100	100
Cluster 3	101	-	-	47.5	0	0	16.8	20.8	14.9	100	0	100	100

Note: *Cluster analysis is based on 30000 series owing to limited computing resources.

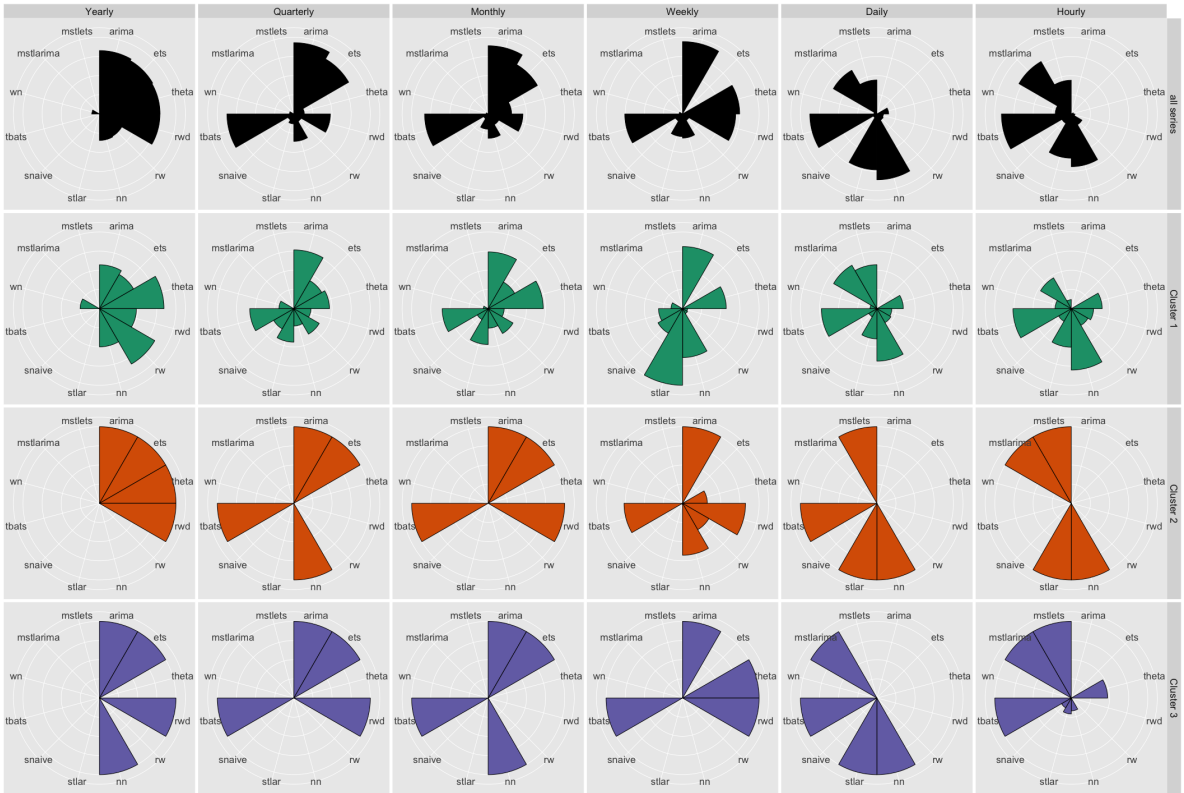

Figure 6: *Visual representation of relative frequencies that each forecast model was selected as a component to the calculation of combination forecasts based on FFORMPP. Each polar coordinate shows relative frequency that each forecast model was selected as a component to the calculation of combination forecasts.*

We now examine the locations of different clusters of M4 competition series in the instance space defined by features. Figure 7 shows the locations of the three clusters in the instance space computed based on the t-SNE approach. For yearly series we can see that most of the series in cluster 1 fall just below the diagonal, while series in clusters 2 and 3 fall within the upper triangular region. Further, clusters 2 and 3 are similar and the series corresponding to cluster 2 and 3 stay close together. For quarterly series, an interesting pattern of clusters can be observed. The time series in cluster 1 fall within the inner circle, while the series in clusters 2 and 3 fall within the second and third outer rings of the circle. This pattern of clusters is called non-spherical. For monthly series, the grouping of the clusters is not obvious. However, a close inspection of the instance space shows a spiral pattern of clusters. Further, cluster 3 is preferred by the series in the upper right corner. For weekly series, clusters are more dispersed across the instance space. For daily and hourly series, clear separation of cluster 1 from clusters 2 and 3 can be observed. Clearly, within each frequency category, similar clusters (for example, clusters 2 and 3 for yearly, quarterly and monthly series) are located close together in the instance space ensuring the similarity of the features of those time series.

The useful description provided by Figure 7 further prompts us to consider the challenge of identifying how the features of time series influence the grouping of these clusters. We now consider how different features vary across the instance space to understand how the locations of different time series reveal the relationship between the features and cluster separation. The preliminary results of the M4 competition (Makridakis, Spiliotis & Assimakopoulos 2018) show that randomness of time series is the most critical factor influencing the forecast accuracy followed by linearity. Further, in their follow-up paper Spiliotis et al. (2019) point out that highly trended and seasonal time series tend to be easier to forecast. The information about remainder series is useful for gaining an idea of the random variation not explained by the trend and seasonality of the series. Hence, we explore the instance space corresponding to the features, strength of trend (`trend`), linearity, strength of seasonality (`seasonality`) and `e_acf1` (the first autocorrelation coefficient of the remainder series after applying STL decomposition on the time series) (Cleveland et al. 1990). The results are shown in Figure 8 - Figure 13 for yearly, quarterly, monthly, weekly, daily and hourly series respectively. Combining the views of the instance space in Figure 7 and Figure 8 - Figure 13 gives us a picture of how the features contribute to the differences in clusters. According to the results of Figure 8- Figure 11, for yearly, quarterly, monthly and weekly series, highly trended series are more likely to fall within clusters 2 and 3. For yearly, quarterly and monthly series, most of the time series in the green cluster (in Figure 7) are less trended with low linearity (in Figure 8 - Figure 10). Hence, those time

series can be considered hard or challenging series to forecast. This is further confirmed from the results of Table 6 and Figure 7: cluster 1 is more heterogeneous according to the way the models are selected to compute combination forecasts (because they use different combinations of individual forecast models), in contrast to clusters 2 and 3. Further, according to the Figure 8-Figure 10, we can see that for yearly, quarterly and monthly series the instance space coloured by linearity exhibits similar structure to the corresponding cluster distribution across instance space shown in Figure 7. For daily series, the features seasonality and e_acf1 clearly separate cluster 1 from the rest. However, for other frequency categories, a clear separation of features with respect to seasonality and e_acf1 cannot be observed. Figure 13 shows that, for hourly time series, features appear to separate the instance space into left and right.

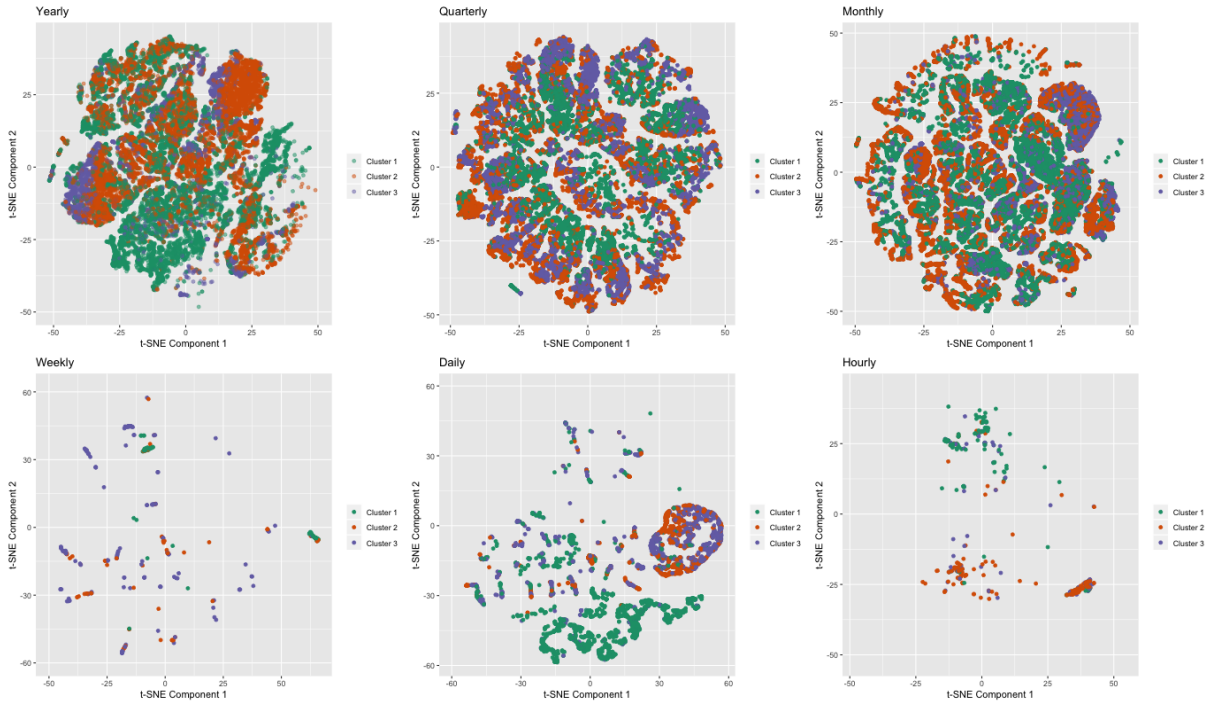

Figure 7: *Location of the three clusters in the instance space. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

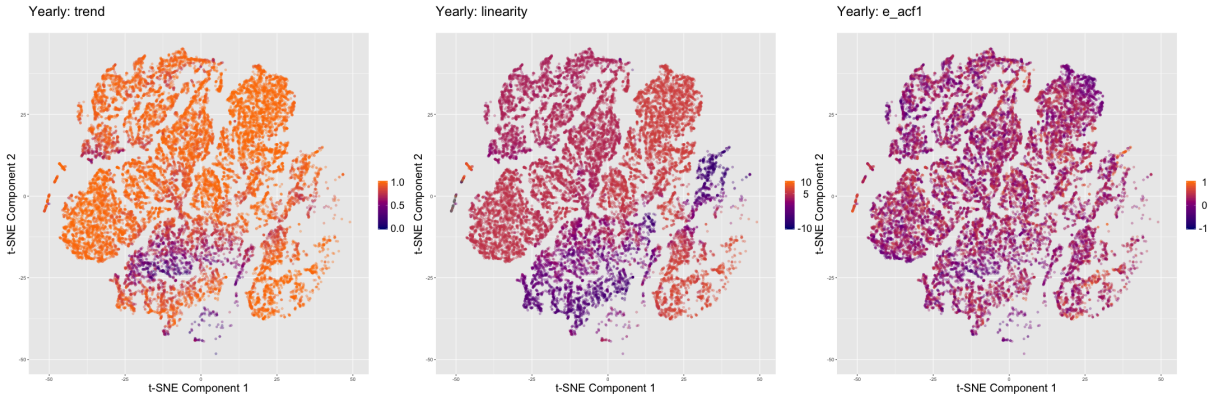

Figure 8: *Distribution of features trend, linearity and e_acf1 across instance space of M4 competition yearly series. Yearly series in cluster 1 (in Figure 7) are mostly less trended and yearly series in cluster 2 are linear and trended. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

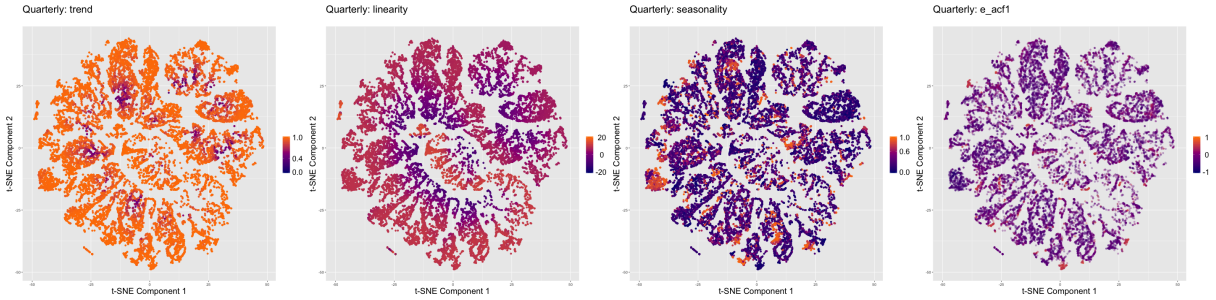

Figure 9: *Distribution of features trend, linearity, seasonality and e_acf1 across instance space of M4 competition quarterly series. Quarterly series in cluster 1 (Figure 7) have low values for linearity. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

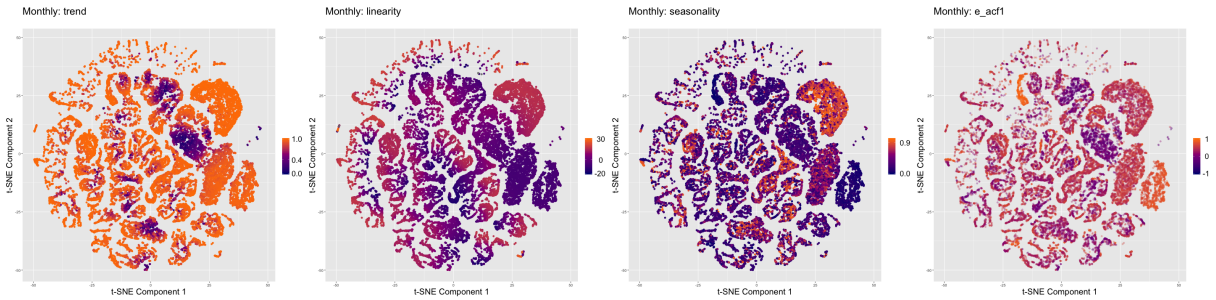

Figure 10: *Distribution of features trend, linearity, seasonality and e_acf1 across instance space of M4 competition monthly series. The locations of monthly time series in cluster 1 (Figure 7) take low values for trend and linearity. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

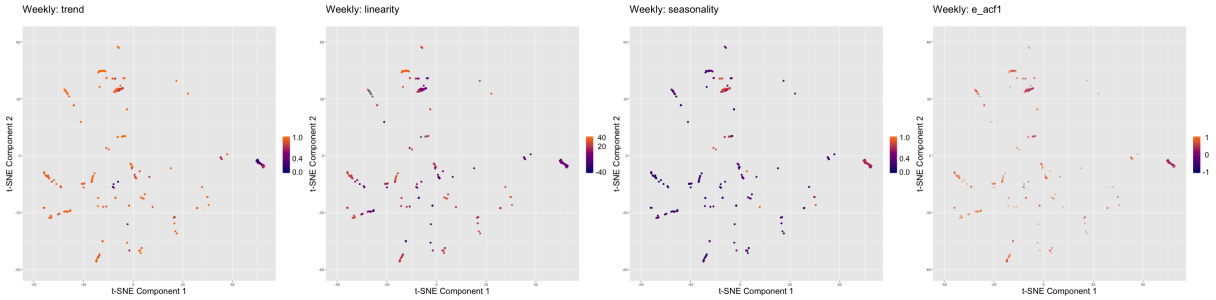

Figure 11: *Distribution of features trend, linearity, seasonality and e_acf1 across instance space of M4 competition weekly series. Most of the highly seasonal time series belong to cluster 1 in Figure 7. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

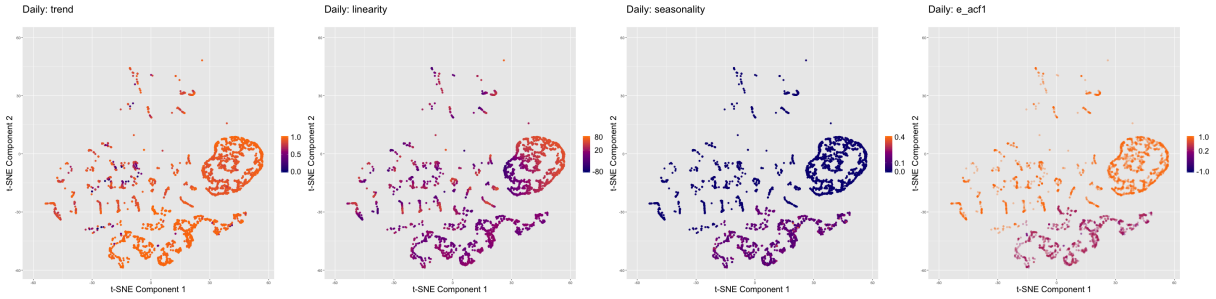

Figure 12: *Distribution of features trend, linearity, seasonality and e_acf1 across instance space of M4 competition daily series. The series in the bottom half of the instance space have high values for strength of seasonality. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

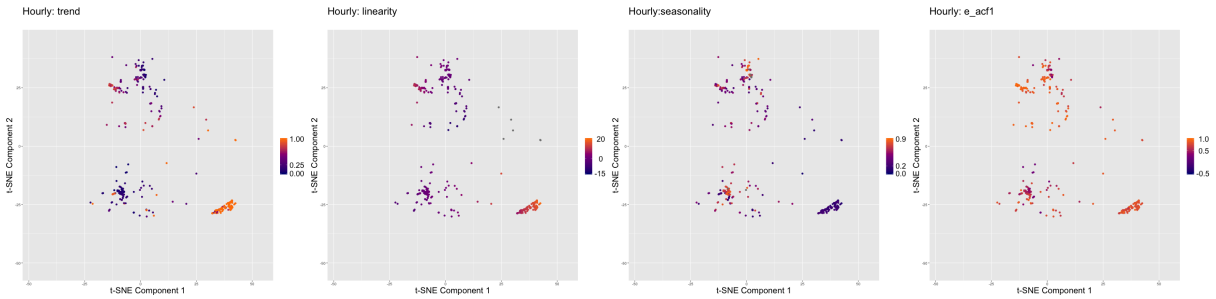

Figure 13: *Distribution of features trend, linearity seasonality and e_acf1 across instance space of M4 competition hourly series. The instances in the lower right corner of the instance space are highly trended and less seasonal. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)*

4 Conclusions

This paper proposes a new meta-learning framework for large-scale time series forecasting. The proposed framework can be used to compute both individual forecasts and combination forecasts. Our results show that features of time series are useful in selecting an optimal subset of models from all individual models without the need to run all possible combinations of

individual models during the online phase. Apart from the obvious utility of this approach for forecast model selection, the ranking of models provides an alternative solution to practitioners who may wish to incorporate their own judgements or expertise into the forecasting decision process. In general, the performance of any model strongly depends on the dataset (reference set) used to train the model. We investigated the feasibility of using the GRATIS approach to increase the diversity of the reference set. This approach is very useful when researchers have a small sample with which to build a reliable classifier, or no sample is available because of data privacy issues. In such circumstances, when applying GRATIS two choices have to be made. This includes: i) the choice of feature values and ii) selection of appropriate values for the parameters of the MAR models. The values for the features can be set by randomly selecting values over the theoretical ranges of features (for example, trend [0, 1]) or based on experts' knowledge in the field of application. For parameters, instead of selecting a single value for each parameter the risk of choosing the wrong one can be reduced by selecting a set of multiple values for each parameter. Hence, it is best to generate a collection of series using multiple values for parameters in order to obtain a more diverse and representative collection of series. For each parameter, a multiple sets of values can be selected from the distributions explained in Kang, Hyndman & Li (2019). We further explored the instance space defined by features to understand how certain features of the time series are influencing the forecast model selection. A further contribution of the paper is provision of empirical support for the findings of the M4 competition (Makridakis, Spiliotis & Assimakopoulos 2018) that hold that the combination forecasts, in general, outperform the best individual forecasts. An interesting future extension of this framework would be to apply this methodology to producing probabilistic forecasts. The FFORMPP framework is implemented in an R package `fformpp`, which can be downloaded from https://github.com/thiyangt/fformpp.

5 Acknowledgements

Thiyanga S. Talagala's research was supported by the Australian Research Council (ARC) the Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS) and the Central University of Finance and Economics, Beijing, China. Feng Li and Yanfei Kang's research were supported by the National Natural Science Foundation of China (No. 11501587 and No. 11701022, respectively). This research was supported in part by the Monash eResearch Centre and eSolutions-Research Support Services through the use of the MonARCH High Power Computing (HPC) Cluster.

Appendix: Correlation plots

Figure 14: *Yearly: Correlation between MASE values across different forecast models for the series in the reference set*

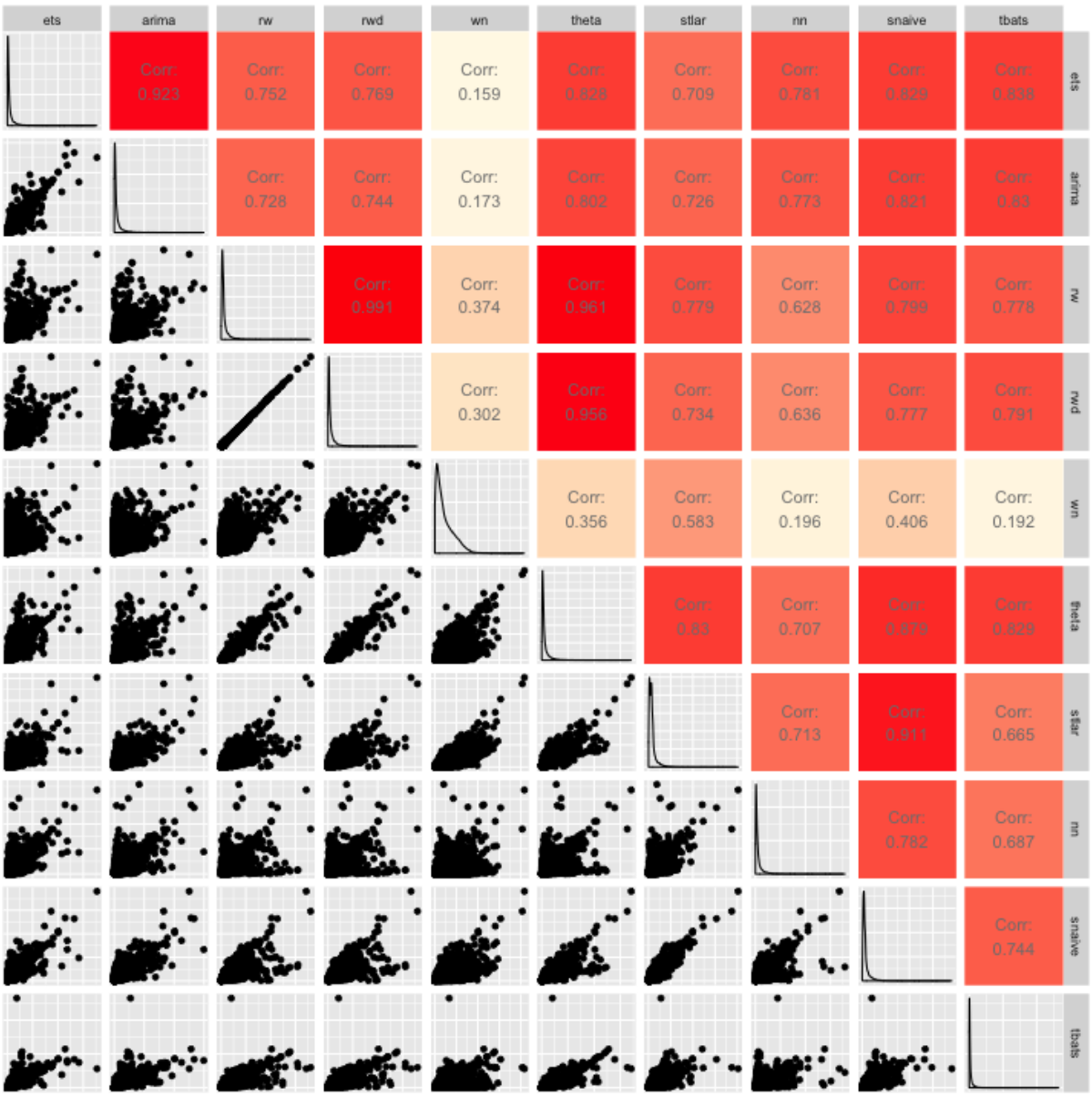

Figure 15: *Quarterly: Correlation between MASE values across different forecast models for the series in the reference set*

Figure 16: *Monthly: Correlation between MASE values across different forecast models for the series in the reference set*

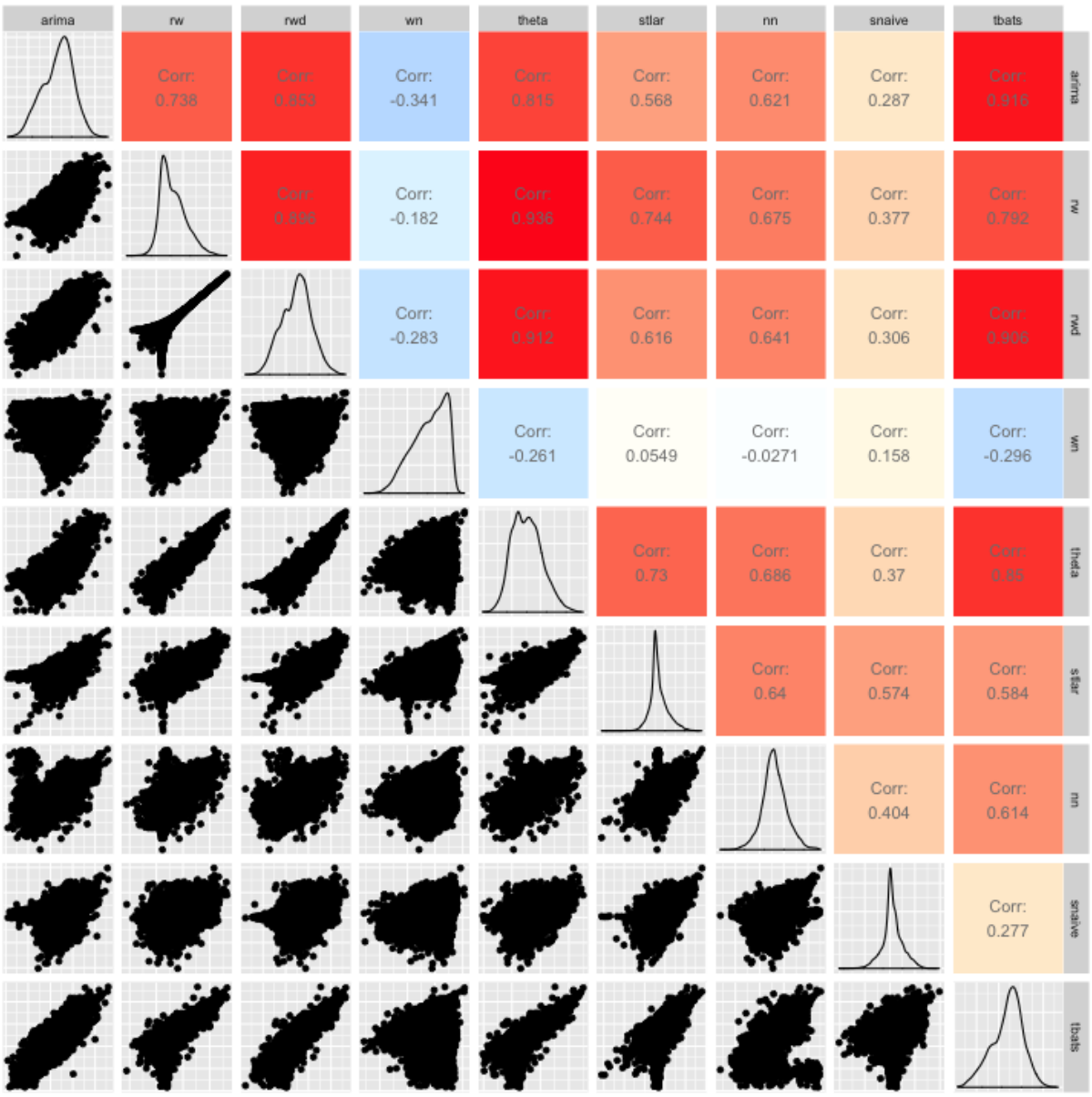

Figure 17: *Weekly: Correlation between MASE values across different forecast models for the series in the reference set*

Figure 18: *Daily: Correlation between MASE values across different forecast models for the series in the reference set*

Figure 19: *Hourly: Correlation between MASE values across different forecast models for the series in the reference set*

References

Adya, M, F Collopy, JS Armstrong & M Kennedy (2001). Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting* **17**(2), 143–157.

Cleveland, RB, WS Cleveland, JE McRae & I Terpenning (1990). STL: a seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* **6**(1), 3–73.

Collopy, F & JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Hyndman, RJ & AB Koehler (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting* **22**(4), 679–688.

Ingel, A, N Shahroudi, M Kängsepp, A Tättar, V Komisarenko & M Kull (2019). Correlated daily time series and forecasting in the M4 competition. *International Journal of Forecasting*.

Kang, Y, RJ Hyndman & F Li (2019). GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. *arXiv preprint arXiv:1903.02787*.

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

Li, F & M Villani (2013). Efficient Bayesian multivariate surface regression. *Scandinavian Journal of Statistics* **40**(4), 706–723.

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

Meade, N (2000). Evidence for the selection of forecasting methods. *Journal of Forecasting* **19**(6), 515–535.

Montero-Manso, P, RJ Hyndman, G Athanasopoulos & TS Talagala (2019). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, [to appear].

Petropoulos, F, S Makridakis, V Assimakopoulos & K Nikolopoulos (2014). 'Horses for Courses' in demand forecasting. *European Journal of Operational Research* **237**(1), 152–163.

Prudêncio, R & T Ludermir (2004). Using machine learning techniques to combine forecasting methods. In: *Australasian Joint Conference on Artificial Intelligence*. Springer, pp.1122–1127.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Spiliotis, E, A Kouloumos, V Assimakopoulos & S Makridakis (2019). Are forecasting competitions data representative of the reality? *International Journal of Forecasting*.

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

Tashman, LJ & ML Leach (1991). *Automatic forecasting software: A survey and evaluation*.

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Chapter 6

Conclusion

6.1 Summary of the main ideas and contributions

Forecasting is a key activity for any business to operate efficiently. Rapid advances in computing technologies have led to organisations being able to collect and access unimaginable amounts of time series data. Hence, many businesses require reliable and efficient ways for forecasting a large number of time series completely automatically. This thesis has presented three forecasting algorithms for large-scale applications. A fundamental aspect of the algorithms is the use of a meta-learning approach to guide a search for forecast model selection. In each of these algorithms, a vector of features calculated on time series becomes the input to the meta-learner. All the algorithms comprise two phases: the offline phase and the online phase. A key advantage of the algorithms is the idea of outsourcing most of the heavy computational work to the offline phase.

The work of this thesis covers four topics: i) time series features, ii) meta-learning, iii) machine learning interpretability and iv) automatic approaches for large-scale time series forecasting.

The main contribution of Chapter 2, was to propose a meta-learning framework to identify the 'best' forecast-model for each individual series. This framework is referred to as FFORMS: Feature-based FORecast-Model Selection. First a set of features that are useful

in identifying suitable forecast models was identified, then extended by adding previously neglected features of time series. The FFORMS framework builds a mapping that relates the features of a time series to the 'best' forecast-model using a random forest. A key advantage of the proposed framework is that the time-consuming process of training a random forest is performed in the offline phase. The online phase involves calculating the features of a time series and using the pre-trained random forest to identify the best forecast model. Hence, generating forecasts only involves the estimation of a single model and computed forecasts based on the estimated model. In doing so, the FFORMS framework fills an important gap in contemporary forecasting practice, with many available models to choose from and with predictions being required extremely fast.

In Chapter 3, the FFORMS framework was extended to handle weekly, daily and hourly series, as was the diversity of models used as class labels. The application of the FFORMS framework to the M4 competition data was analysed. The FFORMS approach yields accurate forecasts comparable to several benchmarks and other commonly used automated approaches for forecasting. The main contribution of Chapter 3 was the exploration of what is happening under the hood of the FFORMS framework, thereby presenting an understanding of what features lead to the different choice of models for forecasting and how different features influence the predicted outcome. This was accomplished using model-agnostic machine learning approaches. The chapter explored **which** features are the most important for the choice of the FFORMS framework, and **where** they are most important: that is, for the overall classification process, or within a specific class of models or a set of multiple classes of models. Further, partial dependency plots were used to visualise **when** and **how** these features link with the prediction outcome of the FFORMS framework. Finally, the chapter explored **when** and **how strongly** features interact with other features. For all features, the displayed relationships of partial dependency plots are consistent with the domain knowledge expectations. This is an important aspect in encouraging people to trust and use the proposed framework effectively.

In Chapter 4, the main contribution was the new meta-learning framework FFORMA (Feature-based FORecast Model Averaging), which allows weights for forecast combinations to be obtained. An extreme boost gradient algorithm with a custom objective function

was used to train a meta-model. Similar to FFORMS, the online part of the algorithm requires calculating the features of a time series and using the pre-trained XGBoost-based meta-learner. In FFORMA, the probabilities of each model being the best are used as weights for computing a combination forecast. FFORMA is slower than FFORMS because the final forecasts of FFORMA is the weighted average of several individual models. The FFORMA approach achieved second position in the M4 competition (Makridakis, Spiliotis, and Assimakopoulos, 2018).

Chapter 5, contributed the use of the efficient Bayesian multivariate surface regression approach to model forecast error as a function of features calculated from the time series. This is termed FFORMPP (Feature-based FORecast Model Performance Prediction). This framework took the correlation structure between forecast errors of different models into the meta-learner training process. FFORMPP allows ranking of the models with respect to their forecast errors and evaluates their relative forecast performance without calculating forecasts from all available individual models in the pool. The rankings of models provide an alternative solution to practitioners who may wish to incorporate their own judgements or expertise into the forecasting process.

One of the special components of the proposed meta-learning frameworks is augmenting the observed sample with simulated time series. This process may be useful when the number of observed time series is too small to build a reliable meta-learner. Alternatively, one might wish to add more of some particular types of time series to the reference set to achieve a more balanced sample. Chapter 2 and Chapter 3 explored a use of the model-based approach to simulate time series. The time series were simulated based on ARIMA and ETS models. The second contribution of Chapter 5 was to examine the feasibility of using a feature-based time series simulation approach in generating a realistic time series collection to obtain a diverse collection of time series. The chapter explored the use of GRATIS (GeneRAting TIme Series) with diverse and controllable characteristics proposed by Kang, Hyndman, and Li (2019).

The third contribution of Chapter 5 was the exploration of the instance space defined by features to understand how certain features of the time series influence the forecast model selection. Previous work by Kang, Hyndman, and Smith-Miles (2017) shows no dense

concentration of instances according to the best forecast model of time series; the locations for which models are the best are scattered throughout the instance space. However, according to the results of Chapter 5, dense regions are visible depending on which model is selected as a components of the calculation of combination forecasts. This also indicates that time series are not amenable to a single best forecast model but to a particular set of individual models. This visualisation of instances also follows Wickham, Cook, and Hofmann (2015)'s philosophy of 'representation of model in the data space (m-in-ds)'. Displaying the data in the model space (d-in-ms) is the most commonly used approach for model diagnostics, for example, a plot of fitted values versus residuals (Wickham, Cook, and Hofmann, 2015). m-in-ds is a visualisation of embedding high-dimensional data into a low-dimensional space generated from the model. Visualisation of m-in-ds helps to gain an understanding of the nature of the relationship between features and predicted outcomes. In the context of classification, representation of m-in-ds could be achieved by first, projecting the training data set into meaningful low-dimensional feature space and second visualising the complete prediction regions or their boundaries. In other words, this can be considered the visualisation of predictor space in the context of data space.

A comparison of point forecasts values based on the three frameworks over the M4 competition data is shown in Table 6.1. For each method, out-of-sample MASE over the forecast horizons was calculated and averaged over all time series. In general, FFORMA, and FFORMPP consistently forecast more accurately than all benchmark methods, except the random walk in daily series. FFORMA and FFORMPP performed equally well for yearly, quarterly and monthly series. For weekly, daily and hourly series, FFORMA provided substantially more accurate forecasts. The FFORMS approach also achieved comparable results in a much more cost- and time-effective manner. It is important to note that FFORMS forecasts are computed based on a single forecasting model (applying individual best forecast model to each series), FFORMA forecasts are computed based on a combination of nine different models and FFORMPP forecasts are computed by taking the median of individual forecasts corresponding to the four models with minimum predicted MASE. Hence, if the focus is to achieve reasonably accurate forecasts in a limited timeline, the FFORMS approach provides a solution. FFORMA is suitable if the

aim is to achieve more accurate forecasts and the time and computing budget is not restricted. If the aim is to obtain reasonably accurate forecast using a reasonable time and computing budget, FFORMPP offers a promising solution for yearly, quarterly and monthly series. The M4 competition winning method, hybrid Exponential Smoothing-Recurrent Neural Network (ES-RNN) approach is a synthesis of exponential smoothing model with advanced Long Short Term Memory (LSTM) neural networks (Smyl, 2019). The power of Slawek's ES-RNN approach lies in the co-training of both the per-time series exponential smoothing model parameters and the general RNN parameters in the deep learning layer of the architecture. According to Table 6.1 FFORMS approach outperformed ES-RNN approach for weekly series and FFORMA outperformed the ES-RNN for both weekly and daily series. Furthermore, a total of six machine learning-based forecasting algorithms (5 submissions, 1 benchmark method based on MLP: multilayer perceptron) were considered in the M4 competition. These algorithms include simple neural network architectures as well as deep learning neural network architectures. All of them were less accurate than FFORMA and 5 of them were less accurate than FFORMS approach (Makridakis, Spiliotis, and Assimakopoulos, 2019) . Furthermore, these machine learning and deep learning forecast models can be easily integrated as part of the algorithm space in FFORMS, FFORMA and FFORMPP.

Table 6.1: *MASE values calculated over the M4 competition data*

Point Forecasts (Mean Absolute Scaled Error (MASE))						
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.17	1.20	0.98	2.31	3.57	0.84
FFORMA	3.06	1.11	0.89	2.11	3.34	0.81
FFORMPP	3.07	1.13	0.89	2.46	3.62	0.96
auto.arima	3.40	1.17	0.93	2.55	-	-
ets	3.44	1.16	0.95	-	-	-
theta	3.37	1.24	0.97	2.64	3.33	1.59
rwd	3.07	1.33	1.18	2.68	3.25	11.45
rw	3.97	1.48	1.21	2.78	3.27	11.60
nn	4.06	1.55	1.14	4.04	3.90	1.09
stlar	-	2.02	1.33	3.15	4.49	1.49
snaive	-	1.66	1.26	2.78	24.46	2.86
tbats	-	1.19	1.05	2.49	3.27	1.30
wn	13.42	6.50	4.11	49.91	38.07	11.68
mstlarima	-	-	-	-	3.84	1.12
mstlets	-	-	-	-	3.73	1.23
combination (mean)	4.09	1.58	1.16	6.96	7.94	3.93
M4-1st	2.98	1.12	0.88	2.36	3.45	0.89

One limitation of the analysis is that it does not report the computational time owing to different platforms used to run the frameworks. However, all three frameworks are

Table 6.2: *Computational time for producing forecasts based on 100 randomly selected series from each frequency category of the M4 data set.*

	Computational time for producing forecasts in seconds (IQR)					
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.38 (0.18)	20.98 (8.23)	100.13 (7.13)	128.41 (5.17)	77.20 (5.67)	34.53 (5.16)
FFORMA	23.41(0.26)	144.39 (0.60)	873.25 (0.32)	718.68 (5.24)	886.33 (6.31)	821.98 (5.42)
FFORMPP	5.31(0.21)	30.45 (1.44)	190.05 (5.15)	183.36 (6.78)	93.76 (0.34)	56.14 (11.67)
auto.arima	5.91 (0.05)	42.11 (2.15)	448.41 (1.95)	584.98 (2.35)	-	-
ets	1.14 (0.02)	16.92 (0.09)	115.50 (0.17)	-	-	-
theta	2.74 (2.48)	10.15 (11.45)	29.13 (1.15)	96.06 (0.42)	83.77 (2.67)	54.32 (2.32)
rwd	0.29 (5.42)	0.29 (8.20)	0.33 (15.57)	0.34 (21.78)	0.41 (33.72)	0.37 (26.46)
rw	0.16 (4.65)	0.20 (6.67)	0.26 (15.68)	0.22 (17.16)	0.27 (19.56)	0.24 (9.58)
nn	2.32 (0.14)	6.54 (0.23)	32.78(0.25)	281.68 (0.61)	424.28 (1.71)	354.97 (3.61)
stlar	-	0.83 (17.97)	0.94 (12.03)	0.90 (10.11)	2.21 (0.12)	1.70 (0.01)
snaive	-	0.18 (4.51)	0.30 (3.12)	0.20 (1.44)	0.32 (0.34)	0.44 (2.12)
tbats	-	20.16 (6.98)	38.12 (2.44)	40.16 (3.36)	68.73 (0.65)	49.52 (2.98)
wn	0.19 (2.65)	0.20 (4.30)	0.23 (0.08)	0.19 (4.51)	0.26 (1.00)	0. 22 (0.05)
mstlarima	-	-	-	-	86.92 (0.52)	30.60 (0.17)
mstlets	-	-	-	-	19.79 (0.09)	10.13 (0.48)

scalable both in time and computing costs. Further, all three frameworks can be easily parallelised for a given computing budget by dividing the process into separate steps. To ensure a fair comparison, computational time for producing forecasts based on 100 randomly selected series from each frequency category of the M4 competition data set is given in Table 6.2. The reported values are median elapsed time of 100 replicates. The corresponding Inter Quartile Ranges (IQRs) are given in parentheses. Table 6.3 reports the computational time for features. None of the features are computationally demanding. The computational time was measured using the R package microbenchmark (Mersmann, 2019) on 24 core Xeon-E5-2680-v3 @ 2.50GHz servers.

According to Table 4 in Chapter 2 and Table 7 in Chapter 5, the most accurate forecast models are the ones that most frequently get selected by the meta-learners. For example, according to Table 6.1 the random walk with drift performs extremely well with yearly series. According to Table 4 in Chapter 2 the random walk with drift has been selected 46.75% of the time by FFORMS and according to Table 7 in Chapter 5 the random walk with drift has been selected as one of the components in calculating combination forecasts for all series by FFORMPP. The interpretations for other frequency categories can be made in a similar fashion. Further, across all frequency categories white noise process was the least selected forecast model and according to Table 6.1 it was the worst performing model across all frequency categories. Since, the white noise process turned out to be the least accurate, it was not considered as a base model to the algorithm space of FFORMA. This

Table 6.3: *Computational time for features over 100 time series.*

Feature category (as in "tsfeatures" package)	Feature	Description	Computational time (median/(IQR))
	T	length of time series	47 (3) nanoseconds
stl_features	trend	strength of trend	862.45 (5.27) milliseconds
	seasonality	strength of seasonality	
	linearity	linearity	
	curvature	curvature	
	spikiness	spikiness	
	e_acf1	first ACF value of remainder series	
	e_acf10	sum of first 10 ACF value of remainder series	
	peak	strength of peak	
	nperiods	number of seasonal periods in the series	
	trough	strength of trough	
	seasonal_period	length of seasonal period	
hurst	hurst	Hurst exponent	83.31 (4.79) milliseconds
stability	stability	stability	80.24 (4.16) milliseconds
lumpiness	lumpiness	lumpiness	154.22 (2.08) milliseconds
entropy	entropy	spectral entropy	247.57 (5.31) milliseconds
nonlinearity	nonlinearity	nonlinearity	255.57 (2.53) milliseconds
holt_parameters	alpha	ETS(A,A,N) $\hat{\alpha}$	367.50 (6.84) milliseconds
	beta	ETS(A,A,N) $\hat{\beta}$	
hw_parameters	hwalpha	ETS(A,A,A) $\hat{\alpha}$	4.89 (0.08) seconds
	hwbeta	ETS(A,A,A) $\hat{\beta}$	
	hwgamma	ETS(A,A,A) $\hat{\gamma}$	
unitroot_pp	ur_pp	test statistic based on Phillips-Perron test	174.44 (1.66) milliseconds
unitroot_kpss	ur_kpss	ur_kpss	66.44 (4.25) milliseconds
acf_features	y_acf1	first ACF value of the original series	255.23 (3.85) milliseconds
	y_acf10	sum of squares of first 10 ACF values of original series	
	diff1y_acf1	first ACF value of the differenced series	
	diff1y_acf10	sum of squares of first 10 ACF values of differenced series	
	diff2y_acf1	first ACF value of the twice-differenced series	
	diff2y_acf10	sum of squares of first 10 ACF values of	
	seas_acf1	autocorrelation coefficient at first seasonal lag	
	lmres_acf1	first ACF value of residual series of linear trend model	50.21 (2.34) milliseconds
pacf_features	y_pacf5	sum of squares of first 5 PACF values of original series	313.39 (5.73) milliseconds
	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	
	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	
	seas_pacf	partial autocorrelation coefficient at first seasonal lag	
crossing_points	crossing_points	number of times the time series crosses the median	26.73 (2.62) milliseconds
arch_stat	ARCH.LM	ARCH LM statistic	135.47 (3.26) milliseconds
heterogeneity	arch_acf	sum of squares of the first 12 autocorrelations of z^2	441.63 (6.04) milliseconds
	garch_acf	sum of squares of the first 12 autocorrelations of r^2	
	arch_r2	R^2 value of an AR model applied to z^2	
	garch_r2	R^2 value of an AR model applied to r^2	

helped to reduce the computing time while maintaining its high accuracy. Hence, prior knowledge about the forecast accuracy of base models can be used to reduce the number of models to be used in the algorithm space. This will intern speed up the both online and offline phases of the classification process without losing too much information.

In our algorithms we use a large sample of time series to create a reference set for training the meta-learners. Since the processing of data to create the reference set is done in the off-line phase of the algorithms, the computational time is of no real consequence. However, it could be the case that the meta-learner should be re-trained once in a while. For example,

that could happen in a retail company where the meta-learner is trained with its own data to forecasts the sales of its own products which features may significantly change over time. In order to get an idea about when and how often the re-training process should be done, two-dimensional instance space, defined by the features could be used. For that, we first compute the principal components projection using the features in the reference set, and then project the new time series to the same low-dimensional feature space. If the new time series fall within the space covered by the series in the reference set a new meta-learner is not required. If any of the series fall out-side of the space covered by the reference set a new-meta learner is required to be trained.

One could argue in the FFORMS algorithm, the random forest probability scores could be used for weighting alternative forecast models and construct a robust combination scheme like FFORMA. However, this approach did not bring any improvement in the performance and furthermore with some series it degraded the performance. The reason is that in FFORMS the forecast model selection problem is treated as a classification problem. Hence, in this case the weights are mostly close to either 0 and 1; the best model has a weight close to 1 and others very close to 0. For example, suppose for a given series the forecast error vector for the random walk, the random walk with drift and a white noise process is [1.31, 1.30, 3.4]. Then, ideally the vector of class weight we expect from FFORMS is [0, 1, 0], i.e. the best model is given class probability 1 and others 0. This limitation motivated us to introduce FFORMA. In forecast-combinations we would expect similar weights for the random walk with drift and the random walk as they both perform equally well. For this, in the objective function of FFORMA, instead of minimizing the classification error we minimize the average forecast error to obtain suitable weights for forecast combinations. In FFORMPP we opt for simplicity. The median of the best four models is used to compute the combination forecasts. As Lichtendahl Jr, Grushka-Cockayne, and Winkler (2013) claim the simpler method of averaging the quantiles, seems to give as good as a result as more elaborate ones. This helps to reduce the computational time while maintaining accuracy.

All our frameworks are robust to outliers present in the time series. Furthermore, with the exception of Spectral Entropy all other features are not affected by missing values.

For the case of calculating Spectral Entropy snaive and naive approaches are used to impute missing values, before computing the feature. However, both the feature space and algorithm space will depend on the specific population of time series models we need to forecast. Hence, a limitation of these frameworks is that expert's knowledge is required to decide the models to be included in the algorithm space and the features to be included in the feature space.

6.2 Software development and research reproducibility

6.2.1 Software

Two R packages were developed based on the frameworks introduced in this thesis.

1. The first R package `seer` is an accompaniment to the framework proposed in Chapter 2. The package is available at https://github.com/thiyangt/seer. To the best of my knowledge `seer` is the first R package to implement the meta-learning framework for time series forecasting.

2. The second R package, `fformpp`, is an accompaniment to the framework proposed in Chapter 5. The package is available at https://github.com/thiyangt/fformpp.

6.2.2 Reproducibility

Peng (2015) writes,

> "Reproducibility is defined as the ability to recompute data analytic results, given an observed data set and knowledge of the data analysis pipeline. Replicability and reproducibility are two foundational characteristics of a successful scientific research enterprise."

Research reproducibility is an important component for research sustainability. The R codes to reproduce all results and figures of each paper are available in the following Github repositories.

Chapter 2: https://github.com/thiyangt/WorkingPaper1

Chapter 3: https://github.com/thiyangt/FFORMSinterpretation. Further, the R package `explainer` contains the main functionalities used to generate partial dependence plots is available at https://github.com/thiyangt/explainer

Chapter 4: https://github.com/robjhyndman/fforma-paper

Chapter 5: https://github.com/thiyangt/chapter5_fformpp

The R codes to reproduce the content of this PhD this are available at https://github.com/thiyangt/PhDThesis

For all papers, `Rmarkdown` was used to produce a readable output file, supported by the `rmarkdown`, `knitr` and `pander` packages. The R package `bookdown` was used to produce a readable output file of this PhD thesis with the support of the Monash University PhD thesis template available at https://github.com/robjhyndman/MonashThesis.

6.3 Future directions

The results clearly demonstrate that features of time series are useful in identifying suitable models for forecasting. There are several directions for future research.

1. *Application to other datasets.* The current applications are limited to M1, M3 and M4 competition datasets. Therefore, the applicability of the proposed frameworks is limited to groups of time series of similar attributes as the data in the M-competitions. For example, the proposed frameworks with the same set of features and forecast models might not be the right choice for forecasting stock return data, or irregular time series, etc. Hence, it is important to expand the frameworks to other datasets that come from different application domains. When adapting the frameworks to other applications the feature space should be revised with appropriate features that measure characteristics of interest. The algorithm space should also be revised with suitable forecast models. A suitable forecast error metric should also be considered to evaluate the performance of different forecast models. For example, retail companies collect a large number of time series related to sales data. Most of these series are intermittent in nature (Seaman, 2018). Low-volume and intermittent time series were not considered in the M-competition (Makridakis, Spiliotis, and Assimakopoulos,

2019). Hence, new features such as proportion of zeros, number of non-zero intervals, kurtosis, etc., need to be selected to the feature space. The use of forecast error metrics such as sMAPE with intermittent series is not also suitable as it would involve division by zero. Furthermore, none of the time series in the M-competition collections have negative values. Hence, these are not representative of stock return series. New features need to be included when adapting the frameworks for such situations. In addition to the new features new forecast model such as ARCH, GARCH, etc. also need to be considered for the algorithm space.

Smith-Miles and Bowly (2015) pointed out the importance of evaluating meta-frameworks using simulated time series with different distributions in the feature space to achieve a better understanding of strengths and weaknesses of algorithms. Hence, in addition to the application of these frameworks to real-world data sets, the GRATIS (Kang, Hyndman, and Li, 2019) approach can be used to create a test bed with controllable features of the instances to evaluate the frameworks. An illustrative example of the idea is shown in Figure 6.1. First the features of yearly M3 series are computed. Principal component analysis is used to project these onto a two-dimensional space referred to as "instance space". The green points correspond to the yearly time series from the M3 competition. The orange points are the target points. The rules learned by a meta-learner trained based on the green points do not claim to be universally applicable, they merely hold for time series with similar feature distribution as M3 data. Hence, to make the meta-learner more generalizable a more diverse collection of time series can be considered that fills and spreads out the instance space. The GRATIS approach can be used to generate new time series from the target points. Note that it is not possible to generate time series from some target points due to natural constraints in feature combinations (Kang, Hyndman, and Smith-Miles, 2017). According to the no free lunch theorem, there is no algorithm that performs best for all kinds of problems. Hence, the application of the frameworks to simulated time series with controllable features will help the understanding of when these frameworks perform well and when they fail. This will provide a valuable insight to improve the generalisability of the frameworks.

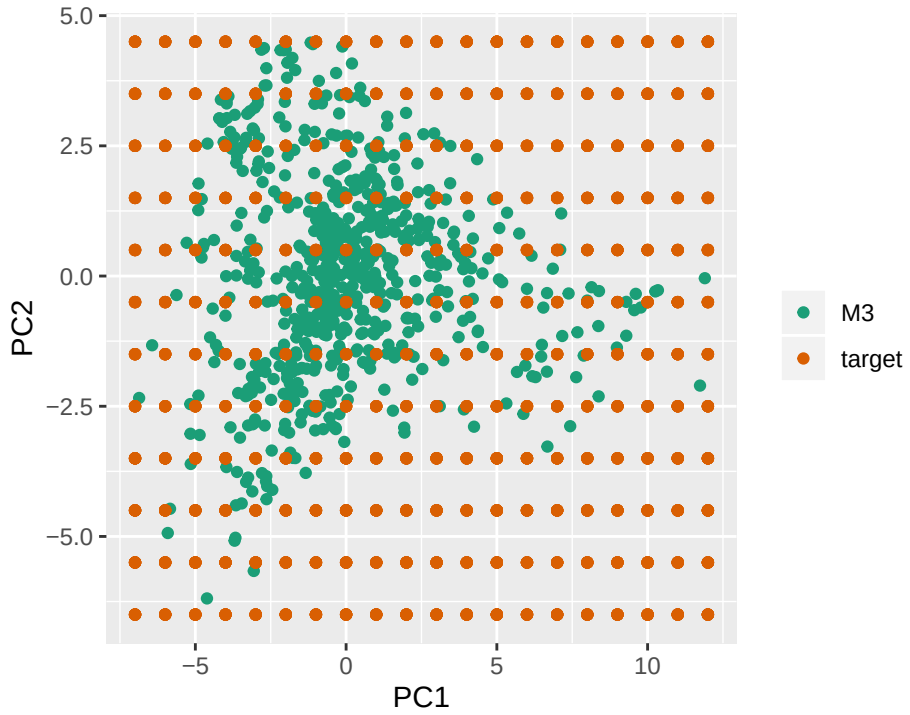

Figure 6.1: *Instance space of yearly M3 time series. PC1 and PC2 are the first two principal components, projected from the features of the yearly M3 time series. The green points are the yearly M3 time series and the orange points are the target points.*

2. *Feature set.* Another important direction for future work is to investigate spectral-domain time series features such as wavelet transform-based features of the time series and spectral density-based features. For example, number of peaks in the global wavelet power spectrum, standard deviation of the wavelet coefficients, etc. It is important to test whether these features would lead to better results for any of the algorithms.

3. *Feature engineering for meta-learning.* The choice of time series features used in the thesis was rather adhoc in nature and developed mostly based on intuition. In Chapter 3, a throughout empirical exploration of features was performed to understand how different features and their interactions affect the choice of forecast model. Identification of a good feature subset is an important component in the meta-learning process, as all our algorithms depend crucially on finding features that enable identification of the best forecast model for the given time series. A good set of features will not only speed up the calculation process, they will help to

obtain even better results. Hence, further research is needed to answer the following contrasting challenges confronting us when designing a meta-learning framework.

3.1: Do we need to add more features? The frameworks introduced in the thesis considered a pool of more than 30 features. However, are these enough? How much information is lost by considering only these features? Is there any possible way to measure the complete information provided by a time series and therefore, to measure the information we miss by using a subset of features to capture it?

3.2: Do we need all the features used in the frameworks? Would it be possible to achieve similar accuracy level by selecting a subset of features?

At the same time, the number of features considered will influence the choice of algorithm used for training. For example, if a single decision tree is considered, then the use of 30 features is probably ineffective as the algorithm is too simplistic to model the connection between forecast model performance and features. However, the algorithms such as random forest, deep-learning architectures can effectively model such complex relationships. Hence, further research is needed to explore if feature engineering process would lead to better results for any of these algorithms in terms of accuracy as well as time.

4. *Meta-learner training process.* An important component of a meta-learning framework is the construction of an engine that maps an input space composed of features to an output space composed of forecast model performance. Another direction to investigate could be to replace the training algorithm with other alternatives such as deep-learning architectures, support vector machines, etc. and test whether these approaches outperform the FFORMS, FFORMA and FFORMPP frameworks.

5. *Probabilistic forecasting.* An interesting extension would be to apply this methodology for producing probabilistic forecasts rather than point forecasts. To account for this, instead of using MASE as the error measure in the offline phase, it could be replaced by a scale-free scoring rule such as log scores and retrain a meta-learner for probabilistic forecasting.

6. *Fast and furious forecasting.* Another strand of research would allow for clustering of time series, with a similar forecasting model being applied to all series within a cluster. Ashouri, Shmueli, and Sin (2019) provides a brief survey of such methods. In this way, the number of models to be estimated can be greatly reduced. However, the approach would lead to a loss of efficiency in using non-optimal parameters, and additional variance might be incurred from potentially selecting a non-optimal forecasting method for a given series. It is important to explore how to balance these effects against the speed improvements that are achieved.

Bibliography

Armstrong, JS (2001). Should we redesign forecasting competitions? *International Journal of Forecasting* **17**(1), 542–543.

Ashouri, M, G Shmueli, and CY Sin (2019). Tree-based methods for clustering time series using domain-relevant attributes. *Journal of Business Analytics* **2**(1), 1–23.

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

Brown, RG (1959). *Statistical forecasting for inventory control*. McGraw/Hill.

Brown, RG (1962). *Smoothing, forecasting and prediction of discrete time series*. Englewood Cliffs: Prentice Hall.

Chen, T and C Guestrin (2016). XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794)*. ACM.

Collopy, F and JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

De Livera, AM, RJ Hyndman, and RD Snyder (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association* **106**(496), 1513–1527.

Fildes, R (1989). Evaluation of aggregate and individual forecast method selection rules. *Management Science* **35**(9), 1056–1065.

Friedman, JH (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 1189–1232.

Fulcher, BD and NS Jones (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Gardner, ES (1985). Exponential Smoothing: The state of the art. *Journal of Forecasting* **4**(1), 1–28.

Georgoff, DM and RG Murdick (1986). Manager's guide to forecasting. *Harvard Business Review*.

Gneiting, T and AE Raftery (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* **102**(477), 359–378.

Guyon, I and A Elisseeff (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research* **3**, 1157–1182.

Hyndman, RJ (2001). It's time to move from what to why. *International Journal of Forecasting* **17**(1), 567–570.

Hyndman, RJ (2019). A brief history of forecasting competitions. *International Journal of Forecasting*.

Hyndman, RJ and Y Khandakar (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* **26**(3), 1–22.

Hyndman, RJ and AB Koehler (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting* **22**(4), 679–688.

Hyndman, RJ, AB Koehler, RD Snyder, and S Grose (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* **18**(3), 439–454.

Hyndman, R, C Bergmeir, G Caceres, M O'Hara-Wild, S Razbash, and E Wang (2018). *forecast: Forecasting functions for time series and linear models*. R package version 8.3. http://pkg.robjhyndman.com/forecast.

Kang, Y, RJ Hyndman, and F Li (2019). GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. *arXiv preprint arXiv:1903.02787*.

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

Kotthoff, L (2014). Algorithm selection for combinatorial search problems: A survey. *AI Magazine* **35**(3), 48–60.

Lawrence, M (2001). Why another study? *International Journal of Forecasting* **17**(1), 574–575.

Li, F and M Villani (2013). Efficient Bayesian multivariate surface regression. *Scandinavian Journal of Statistics* **40**(4), 706–723.

Lichtendahl Jr, KC, Y Grushka-Cockayne, and RL Winkler (2013). Is it better to average probabilities or quantiles? *Management Science* **59**(7), 1594–1611.

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

Makridakis, S and M Hibon (2001). Response to the commentaries on'The M3-Competition: results, conclusions and implications'. *International Journal of Forecasting* **17**(4), 581–584.

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

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

Makridakis, S, E Spiliotis, and V Assimakopoulos (2019). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*.

Meade, N (2000). Evidence for the selection of forecasting methods. *Journal of Forecasting* **19**(6), 515–535.

Mersmann, O (2019). *microbenchmark: Accurate Timing Functions*. R package version 1.4-7. https://CRAN.R-project.org/package=microbenchmark.

Peng, R (2015). The reproducibility crisis in science: A statistical counterattack. *Significance* **12**(3), 30–32.

Petropoulos, F, S Makridakis, V Assimakopoulos, and K Nikolopoulos (2014). 'Horses for courses' in demand forecasting. *European Journal of Operational Research* **237**(1), 152–163.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Seaman, B (2018). Considerations of a retail forecasting practitioner. *International Journal of Forecasting* **34**(4), 822–829.

Smith-Miles, KA (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys* **41**(6), 1–25.

Smith-Miles, K and S Bowly (2015). Generating new test instances by evolving in instance space. *Computers & Operations Research* **63**, 102–113.

Smith-Miles, K and L Lopes (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research* **39**(5), 875–889.

Smyl, S (2019). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*.

Tashman, L (2001). The M3-Competition and forecasting software. *International Journal of Forecasting* **17**(4), 578–580.

Taylor, SJ and B Letham (2018). Forecasting at scale. *The American Statistician* **72**(1), 37–45.

Tukey, JW and PA Tukey (1985). Computer graphics and exploratory data analysis: An introduction. *In Proceedings of the Sixth Annual Conference and Exposition (1985), National Computer Graphics Association, pp 773-785*.

Wickham, H, D Cook, and H Hofmann (2015). Visualizing statistical models: Removing the blindfold. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **8**(4), 203–225.

Wolpert, DH and WG Macready (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.