Meta-learning for Forecasting Model Selection

Sasan Barak

This thesis is submitted for the degree of Doctor of Philosophy

August 2021

Management Science

Abstract

Model selection for time series forecasting is a challenging task for practitioners and academia. There are multiple approaches to address this, ranging from time series analysis using a series of statistical tests, to information criteria or empirical approaches that rely on cross-validated errors. In recent forecasting competitions, meta-learning obtained promising results establishing its place as a model selection alternative. Meta-learning constructs meta-features for each time series and trains a classifier on these to choose the most appropriate forecasting method.

In the first part, this thesis studies the main components of meta-learning and analyses the effect of alternative meta-features, meta-learners, and base forecasters in the final model selection results. We investigate different meta-learners, the use of simple or complex base forecasts, and a large and diverse set of meta-features. Our findings show that stationarity tests, which identify the presence of unit root in time series, and proxies of autoregressive information, which show the strength of serial correlation in a series, have the highest importance for the performance of meta-learning. On the contrary, features related to time series quantiles and other descriptive statistics such as the mean, and the variance exhibit the lowest importance. Furthermore, we observe that using simple base forecasters is more sensitive to the number of groups of features employed as meta-feature and overall had worse performed. In terms of the choice of learners, classifiers with evidence of good performance in the literature resulted in the most accurate meta-learners.

The success of meta-learning largely depends on its building components. The selection and generation of the appropriate meta-features remains a major challenge in meta-learning. In the second part, we propose using Convolutional Neural Networks (CNN) to overcome this. CNN have demonstrated breakthrough accuracy in pattern recognition tasks and can generate features as needed internally, within its layers, without intervention from the modeller. Using CNN, we provide empirical evidence of the efficacy of the approach, against widely accepted forecast selection methods and discuss the advantages and limitations of the proposed approach.

Finally, we provide additional evidence that using meta-learning, for automated model selection, outperformed all of the individual benchmark forecasts.

Keywords: Forecasting; Model selection; Meta-learning; Meta-features; Convolutional neural network.

TABLE OF CONTENTS

1 INTRODUCTION AND BACKGROUND....1
1.1 Motivation....1
1.2 Meta-learning and meta-features....3
1.3 Contribution of the thesis....6
1.4 Structure of the thesis....8

2 META-LEARNING USING STATISTICAL TESTS FOR FORECASTING MODEL SELECTION....9
2.1 Introduction....9
2.2 Model selection in forecasting....11
2.2.1 Forecasting model selection using wrappers....12
2.2.2 Forecast model selection using filters....13
2.2.3 Forecast model selection using machine learning....18
2.3 Features-based representation of time series: meta-features....22
2.4 Experimental Design....27
2.4.1 Methodology....27
2.4.2 Meta-learners for classification....30
2.4.3 Feature encoding....34
2.4.4 Forecasting models as base learners....35
2.4.5 Benchmark dataset....36
2.4.6 Forecasting model selection benchmarks....37
2.4.7 Assessing predictive accuracy....38
2.5 Experimental results....40
2.5.1 Meta-learning versus alternative model selection approaches....40
2.5.2 Meta-learning using "time series tests" versus alternative features....44
2.5.3 Meta-learning robustness by data conditions....48
2.5.4 Limitations in selection accuracy....52
2.6 Conclusion....54

3 ON THE DESIGN OF META-LEARNING FOR FORECAST SELECTION....57
3.1 Introduction....57
3.2 Literature review....60
3.3 Methodology....64
3.3.1 Meta-learning Framework....64
3.3.2 Meta-learners....66
3.3.3 Base forecasters....67
3.3.4 Meta-features....69
3.4 Experimental Design....72
3.4.1 Dataset....72
3.4.2 Base forecasters....73
3.4.3 Meta-learners....74
3.4.4 Meta-features....74
3.4.5 Evaluation metrics....77
3.5 Results....78
3.5.1 Accuracy of base forecasters....78
3.5.2 Meta-learner classification....79
3.5.3 Accuracy of meta-learners....80
3.5.4 Feature importance in Meta-learning....82

3.5.5 Model selection or model averaging ... 84
3.6 Data-driven meta-features ... 86
3.7 Limitations ... 89
3.8 Conclusion ... 91

4 DEEP LEARNING FOR FORECASTING MODEL SELECTION ... 93
4.1 Introduction ... 93
4.2 Background of model selection in time series ... 95
4.2.1 Conventional forecast selection and combination ... 95
4.2.2 Deep learning approaches for time series forecast selection ... 100
4.3 A convolutional neural network for meta-learning ... 100
4.3.1 The convolution layers ... 101
4.3.2 The pooling layer ... 102
4.3.3 The classifier ... 104
4.3.4 Forecast selection and combination ... 106
4.4 Experimental Evaluation ... 106
4.4.1 Datasets ... 106
4.4.2 Convolutional Neural Networks Architecture ... 108
4.4.3 Benchmark Algorithms ... 111
4.4.4 Evaluation scheme ... 111
4.4.5 Model combination ... 113
4.4.6 Python package ... 113
4.5 Results of classification and forecasting with CNN and Benchmarks ... 114
4.5.1 Classification accuracy ... 114
4.5.2 Forecasting errors ... 115
4.6 Discussion ... 120
4.6.1 Stability of the CNN model selection ... 120
4.6.2 Why the CNN works? ... 121
4.7 Conclusion ... 124

5 CONCLUSION ... 126
5.1 Implication for practice ... 129
5.2 Limitations ... 130
5.3 Further Research ... 132

6 REFERENCES ... 135

7 APPENDICES ... 135

APPENDIX A. FEATURES CLASSIFICATION BASED ON THEIR CALCULATION METHOD ... 136

APPENDIX B. TREND TESTS ... 141

APPENDIX C. SEASONALITY TEST ... 142

APPENDIX D. CNN CONFUSION MATRIX ... 143

List of Tables

Table 2.2-1. Summary of meta-learning literature in time series forecasting 21

Table 2.3-1. Summary of implemented meta-features in different studies 26

Table 2.4-1. List of meta-features 35

Table 2.5-1. Classification and Forecasting Results of Model Selection Approaches (80%-20% holdout) 40

Table 2.5-2. Classification and Forecasting Results of Different Meta Features (80%-20% train test holdout) 44

Table 2.5-3. Classification and Forecasting Results of Different Meta-Learning Dataset split 48

Table 2.5-4. The prediction of meta learning for randomly selected time series vs best forecast 54

Table 3.4-1. The classification of M3 data 72

Table 3.5-1. AvgRelMAE of simple base forecasters... 78

Table 3.5-2. AvgRelMAE of complex base forecasters 79

Table 3.5-3. Classification accuracy of meta-learners using simple and complex base forecasters 79

Table 3.5-4. Meta-learning accuracy with simple base forecasters. 80

Table 3.5-5. Meta-learning accuracy with complex base forecasters 81

Table 3.5-6. Importance value of feature groups 82

Table 3.5-7. Comparison of the RelAvgMAE for simple and complex models............. 83

Table 3.5-8. Overall forecasting performance for 12 steps ahead of monthly M4 85

Table 4.4-1. Additive error ETS models 107

Table 4.5-1. Classification accuracy of model selection 114

Table 4.5-2. AvgRelMAE for the 4-model and 15-model samples with 44 and 144 time series length 116

Table 4.5-3. AvgRelAME for the 4-model and 15-model samples with 44 and 144 time series length 116

Table 4.5-4. Overall CNN forecasting results on M3 dataset.......................................117

Table 4.5-5. AvgRelMAE for the 4-model sample by series type...............................118

Table 4.5-6. AvgRelAME for the 4-model sample by series type...............................119

Table 4.5-7. AvgRelMAE for the 15-model sample by series type.............................119

Table 4.5-8. AvgRelAME for the 15-model sample by series type.............................119

Table 4.6-1. Stability of the CNN model selection...120

List of Figures

Figure 2-1. Overview of meta-learning procedure..19

Figure 2-2. Relative error distribution for RF and Xgboost 44

Figure 2-3. Plot of decision tree splitting criteria for Statistical Test and Hyndman, et al. meta-features (80-20% test)...46

Figure 2-4. Decision Tree plots for data splits of 60%-40%, 70%-30%, and 80%-20% (from left to right) 51

Figure 2-5. Plot of 8 time series with complex patterns and structural breaks 53

Figure 3-1. The proposed meta-learning methodology...65

Figure 3-2. AutoEncoder Architecture...71

Figure 3-3. Sample of M4 monthly data..73

Figure 3-4. AvgRelMAE for cases containing different groups of features 84

Figure 3-5. SHAP value for the meta-features importance..86

Figure 3-6. classifying the two variants of the exponential smoothing family without using Auto Encoder...87

Figure 3-7. classifying the two variants of the exponential smoothing family with using Auto Encoder 88

Figure 4-1. Convolutional network structure...104

Figure 4-2. Feature representation of two convolutional layers 121

Figure 4-3. Variation of the exponential smoothing family...122

Figure 4-4. CNN meta space for different channels 124

Acknowledgements

First and foremost, I would like to express my deepest appreciation to Sven F. Crone who provided me the opportunity to join Lancaster University to work with him and never stopped supporting me.

I gratefully acknowledge my second supervisor Prof. Nikolaos Kourentzes, who unconditionally accept my invitation for being part of my research. He always supports my thesis and gives me constructive feedbacks for our all publications.

I should also thank several people within the Management Science department and Centre for Marketing Analytics and Forecasting, including Prof. Robert Fildes and Prof. John Boylan with whom I had insightful discussions. I would also like to thank Ms. Gay Bentinck for being a great supporter of doctoral students. I also gratefully acknowledge the funding support I received from the ESRC which made my PhD possible. I had a great network of friends at Lancaster and to all I am greatly thankful; special mention goes to Dr Oliver Schaer, Dr Ivan Svetunkov, Dr Ramin Raeesi, Carlos Rodriguez Calderon, Amin Yarahmadi, Dr Reza Moghdani, and Amin Jafari. Last but not the least, I would like to thank my family for supporting me throughout my life.

Last but not the least, I would like to thank my family for supporting me throughout my life.

Dr Sasan Barak

07/24/2021

Declaration

This thesis has not been submitted in support of an application for another degree at this or any other university. It is the result of my own work and includes nothing that is the outcome of work done in collaboration except where specifically indicated. Many of the ideas in this thesis were the product of discussion with my supervisor Sven F. Crone and Nikolaos Kourentzes.

Lancaster University, UK

Proudly, I dedicate this thesis to my lovely wife, Masoumeh,

And my dearest daughters, Avinar, and Ava

Who always support me to do my best throughout my academic life.

1 Introduction and Background

1.1 Motivation

The rapid development of computing resources eases the process of collecting a large amount of data for enhancing business analytics. Meanwhile, Big Data brings challenges for forecasting, such as selecting an appropriate forecasting model efficiently for tens of thousands of time series. This has attracted the attention of an increasing number of scholars.

There are multiple approaches used for model selection: ranging from statistical tests to information criteria or empirical approaches that rely on cross-validation (Fildes, 1989, Fildes and Petropoulos, 2015). Information criteria, which are penalised likelihood functions (i.e., Akaike information criteria (AIC), Bayesian information criteria (BIC), and so forth), balance the goodness of fit with model complexity and are calculated based on in-sample data (i.e., data used for fitting the model). In contrast, the cross-validation approach evaluates the models' performance by some error measures (i.e., MASE, RMSE, and so forth) on a validation set, which is not used for fitting the model (Hyndman and Athanasopoulos, 2014). Compared to the information criteria, cross-validation differs in three aspects. First, multi-step-ahead forecasts cán be used to inform model selection. Second, cross-validation can evaluate the forecasts generated by different classes of forecasting models or a combination of different models from various classes. Last, pre-processing of in-sample data, such as truncating and transformations, does not invalidate different models' comparisons. A limitation of the cross-validation approach is that the validation set must be split from the original series, which sometimes may be problematic due to the limited available sample.

However, both these approaches require implementing all candidate forecasting models for each time series before evaluation and selection. These so-called "wrapper approaches" significantly increase computational costs and time when facing a large set of time series. In that case, meta-learning has been proposed as a promising alternative for forecasting model selection and has been explored by many studies. Meta-learning is a "filter" methodology that learns to select the best forecasting model based on extracted time series characteristics (meta-features) without implementing the competing candidate forecasts. Lemke and Gabrys (2010) suggest that meta-learning relates the objective of time series forecasting to selecting or combination of the most suitable forecasting models using a meta-learner based on meta-features extracted from time series. A meta-learner is a classification algorithm, where meta-features are used as input, and the best forecasting model is the label of the classification algorithm. A meta-learner relates the meta-features with models' performance. It outputs a set of weights representing the performance of each candidate model and recommends the most appropriate model (i.e. the one with the largest weight) or a combination of them according to their weights. Meta-learning avoids implementing all candidate forecasting models, which can significantly save computational time and costs.

This doctoral thesis focuses on evaluating alternative building blocks of meta-learning, including meta-learners, base forecasters, and, more importantly, meta-features. To the best of our knowledge, most existing meta-learning methods in forecasting model selection and combination rely heavily on judgementally selected meta-features. Based on a review of previous meta-learning studies in forecasting model selection, a varied set of features are introduced for different situations. Besides, the process of choosing a set of useful manually constructed features is challenging since this process requires a deep understanding of features' concepts and empirical evidence that can demonstrate the usefulness of these features from related studies.

1.2 Meta-learning and meta-features

Although the meta-learning approach has shown its superiority in terms of model selection and combination (Talagala et al., 2018; Montero-Manso et al., 2018), the meta-feature extraction process is challenging and potentially unreliable in the context of Big data (large number of time series). According to Ma and Fildes (2020), current meta-learning studies rely heavily on manually selected meta-features.

Many attempts have been made on mining more useful time series features. Fulcher (2018) summarises thousands of such features to provide more useful insights into the time series structure. Christ et al. (2018) introduce a Python package that can extract 794 features from time series and select statistically significant features via hypothesis tests; they describe the application of these features, such as in regression and clustering. Hyndman et al. (2019) summarise several feature-extracting methods which have been successfully applied in time series problems, including anomaly detection (Hyndman, Wang, and Laptev, 2015), forecasting method evaluation (Kang, Hyndman, and Smith-Miles, 2016), and time series classification (Fulcher and Jones, 2014).

Inspired by these studies related to time series features, we argue that selected time series meta-features play an important role in forecasting using meta learning. Indeed, the potential usefulness of time series features in forecasting has been discussed by some early studies. For example, according to empirical results of M3 Competition (Makridakis and Hibon, 2000), Lawrence (2001) and Hyndman (2001) argue that useful information that exists in time series can improve the process of selecting the most suitable forecaster or a combination of forecasters with appropriate weights. Wang et al. (2009) propose a meta-learning framework to recommend acceptable forecasting models. They introduce a set of comprehensive time series features and visualise the time series features in a two-dimensional map. Due to the generality and usefulness of this vector of features, Widodo and Budi (2013) adopt it to build a meta-learning framework applied in forecasting model selection. They implement Principal Component Analysis (PCA) to reduce the extracted time series features and find that such a dimensional reduction process improves the forecasting performance of meta-learning without damaging its classification performance. Kück et al. (2016) consider a new set of meta-features related to errors of forecasting algorithms and employ Multilayer Perceptron (MLP) as a meta-learner. They evaluate their meta-learning framework in the NN3 time series competition (Crone, Hibon, and Nikolopoulos, 2011) and conclude that including error-based features helps the meta-learner to learn more useful patterns and thus improves the overall accuracy of forecasting.

More recently, Talagala et al. (2018) propose a more general meta-learning framework called FFORMS (Feature-based FORecast Model Selection), where they further expand the feature space to include 33 features based on Wang et al. (2009), Hyndman et al. (2015) and Kang et al. (2017). Montero-Manso et al. (2018) introduce FFORMA (Feature-based FORecast Model Averaging) with 42 meta-features where the output of the meta-learner is a set of weights assigned to each candidate's forecasting models instead of the index of the best forecaster. This approach achieves the second rank in the M4 competition (Makridakis, Spiliotis, and Assimakopoulos, 2020).

Through consideration of the pertinent research in this area, it can be inferred that the selection of a proper subset of meta-features mainly follows the decision makers' expertise which is mostly subjective and non-systematic. Fulcher (2017), Fulcher and Jones (2014) and Timmer et al. (1993) indicate that it is difficult to conclude whether new features presented by researchers are better than existing alternatives. They mentioned that it is hard to determine whether methodologically complicated meta-features outperform simple alternatives. Therefore, analysing the meta-learning framework based on the meta-feature perspective and proposing techniques that rely less on the manually selected features is an urgent need in this field. Besides, having a systematic meta-feature analysing method with the potential for selecting more representative features is also essential for improving the meta-learning performance.

Furthermore, manually extracted meta-features from time series may not reflect all characteristics of the series. According to Li, Kang and Li (2019), current studies in this area mostly employ global features as the input of the meta-learning framework but neglect the importance of some local characteristics which may significantly affect the performance of different forecasting models. For instance, in the area of retail sales, if the promotion or special events exist in some time horizons of a time series, the value of sales in these time horizons is very likely to be significantly different from those without these activities or events (Fildes and Ma, 2019). In this case, if we only use some global features, such as trend or seasonality, the meta-learner may not be able to learn local knowledge, and hence the meta-learner’s input can mislead it. This can be resolved by proposing an automated approach which detects the important meta-features and forecasts based on them simultaneously.

Although previously mentioned studies have stated that their proposed meta-learning frameworks achieve a good result, no study directly argues about the reason for their superiority. It is not clear whether the high accuracy is due to the choice of their proposed features, meta-learners (as classifiers), or alternative forecasting models. Moreover, when there are many time series, the process of manually extracting features will cost comparable time and resources, which lowers the forecasting performance of the whole process of meta-learning particularly when the correlation between these features and forecasting models’ performance is weak.

To the best of our knowledge, there is no study that evaluates the reliability and performance of meta-learning by considering the three facets of meta-features, meta-learners, and base forecasters. Therefore, we aim to analyse these variations in the context of forecasting model selection and combination for business time series.

1.3 Contribution of the thesis

Although there is substantial literature on meta-learning, the investigation of its use on time series data has largely been overlooked from the algorithmic and machine learning viewpoints. In this thesis, we build on this by taking a time series standpoint, bringing the learning from that literature to the specification and use of meta-learning in forecasting.

- First, we propose a series of statistical tests, which build upon time series analysis theory that is often overlooked in the primary machine learning based meta-learning literature and evaluate their effectiveness. We analyse the relationship between these meta-features and the meta-learners and compare the result of meta-learning with individual and aggregate forecasting model selection. We find that statistical tests performed better in meta-learning compared to pre-defined packages such as Tsfeatures.
- Second, we analyse meta-learning from three aspects: meta-features, meta-learners, and base forecasters. To do so, we compare the results of simple and complex forecasting models, as well as single and ensemble meta learners. Moreover, we evaluate different groups of meta-features in the meta-learning context. In terms of the simple and complex base forecasters, we find that using more accurate base forecasters can improve the meta-learning accuracy. Moreover, we show that using more advanced base forecasters overfits the result of meta-learner.
- Third, we evaluate the feature importance of a large number of meta-features within ten groups of statistical tests, statistical description, autoregressive, autocorrelation and so forth, in responding to the forecasting model selection accuracy and analyse the forecasting performance of groups based on variants of meta-learners and base forecasters. We find that the larger Tsfresh set of meta-features performs best, compared to Tsfeatures. However, by eliminating the least important features, we could increase accuracy further.
- Fourth, we propose the MetaTS package, an open-source Python library to ease meta-learning for time series forecasting by offering a

toolkit containing the typical components needed for a meta-learning workflow that facilitates the process of doing it for users. In addition to providing new components and facilities, we aim to unify the available Python libraries which can be useful for meta-learning on time series data.

- Fifth, we run additional experiments using the MetaTS package to investigate the effect of feature importance on meta-learning performance. We show that, first, increasing the number of features does not necessarily improve meta-learning performance, so it may even have an adverse effect. Second, the quality of meta-features is very important in the performance of meta-learning, and features that are data-driven, such as autoencoders, which we extracted using MetaTS, have more quality than pre-defined features.Sixth, we propose a novel deep meta-learning framework where Convolutional Neural Networks (CNNs) can automatically extract and learn useful meta-features from time series. We demonstrate that the proposed CNN can effectively generate relevant features to the base forecasters and reduce the need for ad-hoc meta-features. The proposed CNN is also able to jointly construct the features and the classifier. We find that CNN can improve the meta-learner’s accuracy in the trend and level time series; however, it needs further analysis for the seasonal time series.
- It is worth noting that for chapter 2, we use predefined properties of time series as meta-features, while in chapter 3 and 4 we extracted a new group of meta-features which are fundamentally different from the predefined features. We use autoencoders in chapter 3 and CNN for chapter 4.

- Finally, we evaluate the ability of transfer learning for the deep meta-learner for selecting a forecaster or the combination of forecasters in a real dataset (M3 competition). We compare the performance of our proposed transfer learning framework with conventional model selection approaches and demonstrate its efficacy, but also avenues for future research.

1.4 Structure of the thesis

The next three chapters of this thesis consist of three potential research articles that are either submitted for publication or are in preparation for submission. The following is a brief description of each chapter:

Chapter 2: Meta-Learning Using Statistical Tests for Forecasting Model Selection

In this chapter, statistical tests are presented as a group of feature-based representation of time series, and their effectiveness is evaluated and compared with commonly used meta-features.

Chapter 3: On the Design of Meta-learning for Forecast Selection.

This chapter investigates the building blocks of meta-learning and analyses the impact of meta-features, meta-learners, and base forecasters in the final model selection output. To do so, we investigate three alternative meta-learners, the use of simple or complex base forecasts, and a large and diverse set of meta-features.

Chapter 4: Deep Learning for Forecasting Model Selection

This chapter proposes the use of Convolutional Neural Networks (CNNs) to automatically extract and learn useful and model-performance-relative meta-features from time series and then employ these to classify candidate forecasting models. The deep meta-learner can generate features as needed internally, within its layers, without intervention from the modeller. Its properties and performance are examined using simulated and real data.

Finally, chapter 5 provides a discussion and concluding remarks for the thesis, where current limitations, and future avenues of research are outlined.

2 Meta-Learning Using Statistical Tests for Forecasting Model Selection

2.1 Introduction

Nearly 30 years of research and empirical evidence on identifying and selecting the best forecasting method shows the challenging attribute of the model selection problem. Based on the “No Free Lunch (NFL)” theorem, the criteria of a proper model for one problem may not hold for another problem. This is the crucial idea that no one model works best for every problem and whenever a learning algorithm achieves good results on some problems, it must perform poorly on others (Giraud-Carrier, 2008b). Therefore, building a decision system which can select the best learning algorithm between candidates becomes a worthwhile endeavour. Moreover, the number of modelling algorithms and explanatory variables makes the selecting process a problem in itself to solve. In the business analytics area, finding a true classification or forecasting algorithm along with measuring its parameters is one of the crucial stages which is called 'model selection' (Smith-Miles, 2009).

In the forecasting community, forecasting managers may have to forecast thousands of series every month with heterogeneous time series patterns and many forecasting algorithms to match the series with them. At the same time, the quality and quantity of potential model inputs have increased exponentially, permitting models to use more information sources and support a higher frequency of decision making, such as daily and weekly planning cycles. Consequently, the model selection issue arises in practice as an essential problem across all industries and sectors. All these have facilitated and made necessary an increase in automation of the forecasting model selection.

In business analytics, forecasters mainly use individual approaches such as penalized likelihood methods or empirical cross-validation with different error measures to forecast a collection of series, but a learning process does not exist to make the connection between the data characteristics and the models. To involve the learning process in the model selection, meta-learning from the machine learning field is utilized in the forecasting model selection. Based on Rice (1976), the mathematical definition of the meta-learning system is presented as follows:

For a given problem space $x \in P$ with features $f(x) \in F$, find the selection algorithm $S(f(x))$ in algorithm space A, such that the selected algorithm $a \in A$ maximizes the performance mapping $z(a(x)) \in Z$ in terms of a performance measure p (Rice, 1976). In a meta-learning system, features F from Rice's formulation are called meta-features, and they represent inherent characteristics of a given task, x.

Compared to a base learner (forecasting algorithms) which learns a particular task from the corresponding data (e.g., load forecasting, demand forecasting, and so forth), meta-learning continuously gains knowledge (e.g., the learning algorithm properties, the characteristics of the learning problems) from base learners in order to improve the performance of the learning algorithms (Giraud-Carrier, 2008a). Therefore, two concepts of (i) data characteristics and (ii) the best forecasting model can be connected by meta-modelling, which is a learning approach – seen in, for example, classification algorithms. However, doubts always exist on the efficiency of calculated data characteristics. A common challenge comes with the efficiency of meta-features and how much these could be beneficial in selecting the best model. Is it possible to have less sophisticated but more descriptive meta-features in a meta-learning approach? Is using a complicated selection process in comparison to simple individual selection worth? While to answer these questions, an overview of meta-learning as a model selection procedure as well as the implemented meta-features in forecasting problems is needed.

Therefore, in this chapter, a brief review of the different model selection procedures in time series forecasting is studied, and then meta-learning studies in forecasting model selection are reviewed. More specifically, the implemented meta-feature in the feature-based representation of time series is considered broadly, and the gaps in utilized meta-features are investigated. In this regard, for the first time, the effectiveness of using statistical tests as a new group of meta-features is evaluated and compared with the commonly used meta-features. Finally, the relationship between the applied meta-features and the meta-learner algorithms have been investigated, and the forecasting performance of meta-learning in comparison with individual model selection is further demonstrated.

Consequently, contributions of the current research could be noted as: 1. Empirically demonstrating the usefulness of statistical tests as a new group of meta-features and comparing the forecasting performance with the other group of meta-features 2. Comparing different filter and wrapper model selection approaches and 3. Assessing the forecasting performance of using meta-learning for model selection in different sample sizes.

The structure of this chapter is undertaken as follows: Section 2.2 articulates different model selection approaches in forecasting. Section 2.3 elucidates the features-based representation of time series while Section 2.4 explains the methodology and experimental design. Section 2.5 presents the results of the research and finally, section 2.6 concludes the article, and suggests the further scope of research.

2.2 Model selection in forecasting

Selecting an appropriate forecasting model because of the increasing number of algorithms, large number of business time series, and different time series patterns has turned into a complicated problem (Fildes, 2001).

2.2.1 Forecasting model selection using wrappers

Two conventional approaches in time series forecasting are distinguished. Firstly, “aggregate selection”, which is the implementation of one model for all the time series. Secondly, “individual selection”, which is the identification of one particular method appropriate for each series and then the application of that model for forecasting (Fildes, 1989).

Although the aggregate selection is simple in practice, in principle, different individual models, which take into the account time series characteristics such as trend and seasonality, make the forecasting more accurate (Fildes and Petropoulos, 2015). Further, in consideration of the NFL theorem, there is no guarantee that one forecasting model has proper accuracy in all or even more than one-time series (Ma and Fildes 2020). Fildes (2001) indicated that if individual model selection could be done perfectly, then the gains would be substantial. Individual selection mainly have classified into theoretical and empirical methods (Fildes and Petropoulos, 2015).

A significant characteristic of empirical models is in using a validation (out of sample) performance to evaluate how well a model performs on the part of data which is not considered when fitting the model. Model selection on (cross-) validation has two advantages. Firstly, the performance of multiple step-ahead forecasts can be used to inform selection. Secondly, the validation approach is able to evaluate forecasts derived from any process (including combinations of forecasts from various models). The disadvantage of this approach is that it requires setting aside a validation set, which may not always be feasible.

The theoretical models generally are penalized likelihood functions calculated in the in-sample data (Hyndman and Athanasopoulos, 2014). Two best-known criteria are the AIC and BIC and different developed versions of these two approaches. The robustness of theoretical models is that there is no need for a validation sample which requires more data for model fitting. The downside, however, is that manipulating time series by changing the number of data points or transforming the sample invalidates the comparison. Hence, researchers mainly recommend using them only within a single model family (e.g. between exponential smoothing models) (Kourentzes, 2017). Information criteria measures the accuracy of one step ahead in-sample fitting; however, Fildes and Petropoulos (2015) shows the inefficiency of one step ahead forecasting in model selection. This is because of the fact that the likelihood function cannot be proper for multiple-step forecast if the postulated forecasting error is wrong (Xia and Tong, 2011). Therefore, cross validation statistics are mainly used in this study whilst the results are compared with AIC and BIC as benchmarks.

All of the mentioned model selection approaches, either using information theoretic criteria or empirical criteria, require candidate forecasting algorithms to have been computed and the error utilized for the selection of the best algorithm. In machine learning, this is called the “wrapper approach” (Barak et al., 2015). Therefore, individual selection must implement all candidate models in the whole dataset to evaluate model performance, then select between them.

2.2.2 Forecast model selection using filters

Substantially increasing the number of time series for business forecasting makes the wrapper model selection highly time demanding, and in some cases inefficient. The filter approach is a favourable alternative which extracts information from the time series based on some protocols and exploits it for business forecasting. A filter-based model selection provides patterns that identify which model should be used for each time series forecasting without measuring all the past forecasts in a whole time series.

In the forecasting model selection literature, filter approach-based selection protocols such as variance analysis (Gardner Jr and McKenzie, 1988), automatic identification (Vokurka et al., 1996), and rules-based forecasting (Adya et al., 2001) measure data characteristics and use them in forecasting models to generate the best prediction (Fildes et al., 2007).

In contrast to forecasting model selection using wrappers, practitioners in forecasting as well as statistical and econometric research employ filters to conduct ex-ante model selection, narrowing the number of relevant algorithms based upon time series features identified using visualisation of time series data and statistical tests.

Surprisingly little research has been conducted on how to utilise data visualisation for filtering and model selection. Typically, a forecasting expert employs the use of time series graphs, seasonal diagrams, autocorrelation (ACF) and partial autocorrelation functions (PACF), spectral analysis and various seasonal subseries plots etc. for this purpose (Petropoulos et al., 2018).

In contrast, a number of statistical tests have been developed to determine an adequate model form ex ante by identifying the nature of the underlying data generating process, including statistical tests for stationarity, seasonality, trend, outliers, structural breaks and other regular and irregular time series patterns. As these tests are later on employed as both direct filters and meta-features to train a meta-learning algorithm, based on the literature of time series we introduce a number of more common tests which have been implemented in many studies (Pohlert, 2016).

- Trend Tests:

In testing the trend, the non-parametric *Mann-Kendall Trend* test which has a null hypothesis of "no trend", estimating

$$S=\sum_{k=1}^{n-1}\sum_{j=k+1}^{n}\operatorname{sgn}\left(X_j-X_k\right) \tag{2-1}$$

with $sgn(x)=\begin{cases}1 & if \quad X>0\\ 0 & if \quad X=0\\ -1 & if \quad X<0\end{cases}$

The statistic S is closely related to Kendall's τ as given by τ = S / D, where

$$D=\left[\frac{1}{2}n\left(n-1\right)-\frac{1}{2}\sum_{j=1}^{p}t_j\left(t_j-1\right)\right]^{1/2}\left[\frac{1}{2}n\left(n-1\right)\right]^{1/2} \tag{2-2}$$

p is the number of the tied groups in the data set, and t_j is the number of data points in the jth tied group (Pohlert, 2016).

The *Cox-Stuart* trend test defines a binomial distribution for trend detection. Given a set of ordered observations *X1, X2, ..., Xn*, let *c* = *n*/2 if *n* even, or let c = (*n*+1)/2 if *n* odd. Then pair the data as *X1, X1+c, X2, X2+c, ..., Xn-c, Xn*. The Cox-Stuart test is then simply a sign test applied to this paired data.

Alternatively, *Spearman's rho* (SR) test utilises rank-based non-parametric statistical test for detecting a monotonic time trend in a time series (Lehman, 2005).

- Stationarity Tests:

Next, we introduce a number of stationarity tests, in essence testing the absence of trends and other patterns, most notably the ADF test and KPSS tests.

The *Augmented Dickey-Fuller test* (ADF) test with the null hypothesis of a unit root: The alternative hypothesis could be different depending on which version of the test is used, but it is usually stationarity or trend-stationarity. Therefore, in this chapter, three kinds of tests which are "no constant-no trend", "constant-no trend", and "constant-trend" were established. The ADF test is based on estimating the test regression

$$\Delta y_t = \boldsymbol{\beta}'\mathbf{D}_t + \pi y_{t-1} + \sum_{j=1}^{p} \psi_j \Delta y_{t-j} + \varepsilon_t \tag{2-3}$$

where D_t is a vector of deterministic terms (constant, trend etc.) (Said and Dickey, 1984). The *p* lagged difference terms, Δy_{t-j}, are used to approximate the ARMA structure of the errors, and the value of p is set so that the error εt is serially uncorrelated. Under the null hypothesis, Δy_t is I(0) (no trend) which implies that π = 0. The ADF t-statistic is then the usual t-statistic for testing π = 0. However, as mentioned before, with changing the π = φ−1, we can test the I(1).

As an alternative to the ADF- family of tests, the *Kwiatkowski–Phillips–Schmidt–Shin* (KPSS) tests for testing a null hypothesis that an observable time series has a unit root against the alternative of a stationarity around a deterministic trend (i.e., trend-stationary).

The KPSS test statistic is the Lagrange multiplier (LM) or score statistic for testing $\sigma^2_\varepsilon = 0$ against the alternative that $\sigma^2_\varepsilon > 0$ and is given by

$$KPSS = \left(T^{-2}\sum_{t=1}^{T}\hat{S}_t^2\right)/\hat{\lambda}^2 \tag{2-4}$$

where $\hat{S}_t = \sum_{j=1}^{t}\hat{u}_t$, $\hat{u}_t$ is the residual of regression of y_t on D_t $(y_t = B'D_t + \mu_t + u_t)$ and $\hat{\lambda}^2$ is a consistent estimate of the long-run variance of u_t using $\hat{u}_t$. Under the null, y_t has I(0) (Pohlert, 2016).

- Seasonality Test:

Moreover, we introduce a number of seasonality tests, which include the *Chi square* test, *Friedman* test, Multiplicative seasonality test (*Pearson*), and *Kruskall-Wallis* test.

The *Chi square* test (χ^2) is a goodness-of-fit test for detecting the seasonality which is relatively popular because of its simple mathematical theory (Nwogu et al., 2016). For testing the seasonality, the frequency O_i, i = 1, 2,…, k and the frequency E_i, i = 1, 2,…, k are the observed and expected value frequency at the ith season, respectively. Under the null hypothesis that there is no seasonal effect, then $E_1 = E_2 = \ldots = E_k$ and the statistic

$$T = \sum_{i=1}^{k}\left(\frac{(O_i - E_i)^2}{E_i}\right) \tag{2-5}$$

is asymptotically distributed as χ^2 with $\nu = k - 1$ degrees of freedom (Horn, 1977).

The *Friedman* test calculate the p-value for testing the seasonality as a non-parametric alternative to ANOVA with repeated measures. No normality assumption is required. The Friedman statistic Q is given by

$$Q = \frac{12}{K(K+1)} SS'_{col} \tag{2-6}$$

where SS'_{col} is the sum of squares between groups using the ranks instead of raw data. When $k \geq 5$, the probability distribution of Q can be approximated by that of a chi-squared distribution and the null hypothesis is rejected when $Q > \chi^2_{crit}$.

Furthermore, *Pearson* correlation facilitates a Multiplicative seasonality test (Lehman, 2005).

The *Kruskal-Wallis* test of seasonality is a non-parametric test which is used in place of a one-way ANOVA. *Kruskal-Wallis* statistics H is given by

$$H = \frac{12}{n(n+1)} SS'_B \tag{2-7}$$

where SS'_B is the sum of squares between groups using the ranks instead of raw data. This is based on the fact that $\frac{12(k-1)}{n(n+1)}$ is the expected value (i.e. mean) of the distribution of SS'_B. If there are small sample sizes and many ties, a corrected *Kruskal-Wallis* test statistic $H' = H/T$ gives better results where $T = 1 - \frac{1}{n^3 - n} \sum (f^3 - f)$. Here, the sum is taken over all scores where ties exist, and f is the number of ties at that level.

- Other Combinations:

Finally, we test stationarity (level), trend, and seasonality of time series with some properties of the ACF plot. Slowly decay of ACF plot indicates a trend and the seasonality can be captured with the peak in a seasonal basis. The noise data (level) with ACF of a sampling distribution can be approximated by a normal distribution with mean zero and standard error $\frac{1}{\sqrt{n}}$ where n is the number of observations in the series. We use this info to develop tests of hypotheses and confidence intervals for the ACF. We expect 95% of all sample ACF to be within $\pm z\frac{1}{\sqrt{n}}$, then we have a stationary time series, otherwise the series can have trend / seasonality.

Some of our meta-features are non-parametric tests which do not rely on standard normality assumptions and are often based on ranks rather than raw data. It is possible to extend our feature space by adding more statistical tests; however, the most popular tests have been used in this study, and we would like to investigate explicitly whether or not the meta-learning procedure can capture the correct forecast algorithm with common statistical tests (without measuring the forecast errors).

2.2.3 Forecast model selection using machine learning

Rice (1976) explained the model selection problem as a learning problem which attempts to capture the structural characteristics of the problem or instance and to use them for selection of the most relevant algorithm with a meta-level learning algorithm. In the forecasting, the meta-learning process is a learning approach which selects the best forecasting model based on the features of the time series without implementation of the candidate algorithms at first. Therefore, this is a filter approach with less computational cost for implementation. Figure 2-1 illustrated three main components of the meta-learning procedure including base level models (forecasting algorithms), meta-features, and meta-level models (meta-learners) and their connection.

Using model selection with machine learning in forecasting was first proposed by Arinze (1994). Chu and Widjaja (1994) proposed a neural network system to select among several exponential smoothing models using the autocorrelations. Both of these studies treat model selection as a classification problem, while outputs of the classification algorithm (machine learning model) rank the forecasting models, and the inputs are features calculated from the case study data. Therefore, these works can be viewed as particular examples of meta-learning, but they do not mention the context of meta-learning. Prudêncio and Ludermir (2004) claimed that they used the context of meta-learning in time series forecasting for the first time. Two case studies were implemented, while in the first example, J.48 (a version of the C4.5 decision tree) is utilized to choose among two models of forecasting stationary time series (simple exponential smoothing model (SES), and the time-delay neural network (TDNN)), and in the second example, NOEMON as a meta-learner generates ranks for the pool of forecasting models.

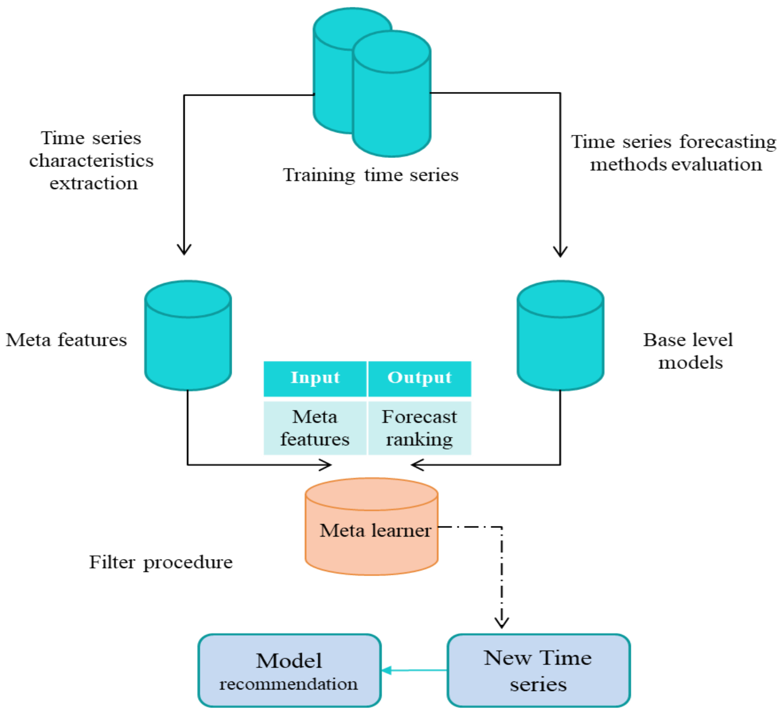


Figure 2-1. Overview of meta-learning procedure

Zhou et al. (2012) proposed an improved decomposition method and back-propagation neural network model (BPNN) for gold price forecasting. In this approach, characteristics of different decomposed subsets of series are explored by training different BPNNs. A rate-based meta-learning model is used to identify the most suitable BPNN model for selecting one of the networks in determining the future trend of gold prices.

Feature selection in meta-learning can be implicitly applied when decision trees are used as the meta-learner; however, within different studies in applying meta-learning to forecasting, to the best of our knowledge, Cui et al. (2016a) is the first work which implemented feature selection methods for meta-features' reduction. They applied singular value decomposition, stepwise regression, and ReliefF for feature selection, and used hit ratio and Spearman’s ranking correlation coefficient as performance evaluation of the meta-learning system. Talagala et al. (2018a) proposed a meta learning framework called FFORMS (Feature-based FORecast Model Selection) that uses a set of 33 meta-features for seasonal and non-seasonal data and a Random Forest as meta-learner to select the best single model from five main forecasting models including, ARIMA, ETS, Random walk, naïve, Theta, and their variants. Montero-Manso et al. (2020) proposed FFORMA (Feature-based FORecast Model Averaging) which is a weighted combined model selection using meta-learning and obtained the second rank in the M4 forecasting competition. The main difference of this study is that they used a weighted combination of Arima, ETS, Theta, Naïve, Seasonal Naïve, Neural Network, Random walk, TBATS, and STLM-AR models instead of solely selection with meta learners. They used XGboost as meta learner with 42 meta-features. All of the features have been previously used in Talagala et al. (2018a) and Montero-Manso et al. (2018), and an R package ‘tsfeatures’ is developed by Hyndman et al. (2019) to facilitate the feature calculation. Recently, Ma, and Fildes (2021) present a meta-learning framework based on newly developed deep convolutional neural networks, which can first learn a feature representation from raw sales time series automatically, and then link the learned features with a set of weights that are used to combine a pool of base-forecasting methods. The experiments which are based on IRI weekly data show that the proposed meta-learner provides superior forecasting performance compared with a number of state-of-art benchmarks, though the accuracy gains over some more sophisticated meta-ensemble benchmarks are modest, and the learned features lack interpretability. When designing a meta-learner for forecasting retail sales, they recommend building a pool of base-forecasters including both individual and pooled forecasting methods, and target finding the best combination forecasts instead of the best individual method.

A comparison between meta-learning studies in forecasting is presented in Table 2.2-1. Applied meta-learner, data set, the domain of the data, forecasting algorithms, feature selection procedure, and findings from the papers are considered as different criteria in Table 2.2-1. By evaluating the last row of Table 2.2-1 (Findings), it can be inferred that meta-learning does not always deliver the minimum error in all cases. However, in addition to the accuracy, the computational time for model selection is also essential. Meta-learning mostly achieves a significant improvement in computational time in comparison to the wrapper model selection approach (Prudêncio et al., 2011). For example, dos Santos et al. (2004) applied random walk, Holt, and auto-regressive (AR) models as sub learners and zoom ranked algorithm as a meta-learner. The results indicated that when the accuracy of the selected model is considered, the zoom rank results are not promising and the Holt model has the lowest error, but when the computation time is added into the cost function, zooming provides a small improvement in the model selection's accuracy. The same evidence was reported by Lemke and Gabrys (2010a) while they considered 15 simple and combined models as base learners, whilst support vector machine, neural network, and decision tree were considered as meta-learning. In their first experiment, none of the applied meta-learners decreased the overall forecasting error; however, the zoom rank algorithm (combination of meta-learners) significantly reduced the symmetric Mean Absolute Percentage Error (sMAPE) of the overall forecasting error.

Table 2.2-1. Summary of meta-learning literature in time series forecasting

	Arinze (1994)	Prudêncio and Ludermir (2004)	Wang et al. (2009)	Lemke and Gabrys (2010a)	Matijaš et al. (2013)	Kück et al. (2016b)	Cui et al. (2016a)	Talagala, et al. (2018)	Montero-Manso, et al. (2020)	Ma, and Fildes (2021)
Meta-learner	DT	1) J48 2) NOEMON	SOM, DT	NN, DT, SVM, Zoomed ranking	Euclidean distance, CART, LVQ network, MLP, Auto MLP, e-SVM, Gaussian Process (GP).	NN	k-nearest neighbour, ANN	Random Forest	XgBoost	CNN
Data set	67-time series	M3 (3003-time series)	315-time series	NN3 111 time series, NN5, 111-time series	1 time series with 69 Task	NN3, 111-time series	44 benchmarks from IEEE CEC 2013&2014	M3	M4	IRI dataset (Bronnenberg , Kruger, & Mela, 2008)
Data Domain	Econometric data on the US economy	Demographic, Finance, Industry, Macro, Micro, Other	economics, medical, engineering	Business, cash machine withdrawal	Load and electricity	Business	Simulated data in Engineering	Business, Finance, Industry, Macro, Micro, Other	Business, Finance, Industry, Macro, Micro, Other	Business
forecast	6 Arima and ES models	1) SES, TDNN	RW, ARIMA, ES, NN	Arima, SES, MA, Tylor, Regression,	RW, ARMA, Similar Days algorithm, Layer RNN,	ANN, ANA, AAN, AAA	kriging, SVR, RBF, MARS, NN, PR	Arima, ETS, Theta, Naïve, RW	Arima, ETS, Theta, Naïve, NN, RW,	ETS, ADL, ARX, ELM, SVM, GBRT,

	Arinze (1994)	Prudêncio and Ludermir (2004)	Wang et al. (2009)	Lemke and Gabrys (2010a)	Matijaš et al. (2013)	Kück et al. (2016b)	Cui et al. (2016a)	Talagala, et al. (2018)	Montero-Manso, et al. (2020)	Ma, and Fildes (2021)
		2) RW, Holt, AR		Theta, NN, Elman NN	MLP, m-SVR, Robust LS-SVM				TBATS STLM-AR	ADLP, RF, ELMP
Feature selection	Implicitly with DT	-	Implicitly with DT	Implicitly with DT and ZR	Feature weighting	-	singular value decomposition, step-wise regression, and ReliefF	-	Feature weighting	Feature weighting
Finding	Faster and more accurate	Faster but with the lower accuracy	Faster and more accurate	Winner of the competition, more accurate with zoom ranking	More accurate	More accurate	More accurate	More accurate	More accurate	More accurate

DT: Decision Tree, RW: Random walk, HL: Holt exponential smoothing, AR: Auto-regressive model, SVR: Support vector regression, RBF: Radial basis function, MARS: Multivariate adaptive regression splines, NN: Neural network, PR: Polynomial regression.

Combinations of meta-learners have also been studied by Matijaš et al. (2013). They considered four types of load forecasting task to apply the meta-learning approach to multivariate time series by ten different base learners and eight meta-learners. The results indicated a lower MAPE for all of the tasks with ensemble of meta-learning classifiers. Moreover, their overall runtime decreased by almost three times in comparison with the wrapper model selection.

After the general overview about model selection and meta-learning, in the next section, feature-based representation of time series will be explained, and a common challenge with the efficiency of meta-features has been addressed.

2.3 Features-based representation of time series: meta-features

Analysing time series based on the property and structure of extracted features enhances our interpretation about the problem and eases the decision-making process for selecting the appropriate forecasting model. The feature-based representation of time series (meta-features) can exploit machine learning potential in discovering the hidden pattern in complex time series.

Feature-based characterization of time series not only can be utilized on univariate time series, but also for multivariate versions, unordered data sets, and features of inter-relation between the pairs of time series. This study mainly focuses on univariate time series to capture the underlying pattern and interesting structures in business domain datasets.

Proper detection of time series' similarities by their features' characterization can tackle time series data mining problems, such as anomaly detection, motif discovery, model selection, clustering, and classification of time series (Bagnall et al., 2017). The superiority of using meta-features in these problems is that meta-features can take full-time series as an input rather than shorter subsequences, distilling the complex pattern into understandable, low dimensional properties. So, meta-features can be implemented on the time series with different lengths, phase-alignments, and context of the problem; the outcome will then be a matrix of time series (row) × features (columns) which is similar to the statistical learning cases. Therefore, there is no problem related to the length of the time-series and alignment in time. Secondly, the current space of the time series may not mirror the problem characterization, while a well-suited meta-feature transfers the problem into the feature space and provides interpretability and better understanding about the problem (Harvey and Todd, 2015). Note that different authors have discussed that the critical issue does not lie in developing or finding a complex prediction/classification algorithm, but in the selection of well-discriminating features (Timmer et al., 1993, Bagnall et al., 2012). Based on evidences from the Kaggle competition, the significant difference between the leaders and other competitors is hardly implementing sophisticated models, but creating relevant features from the dataset and predicting with them (Blum and Hardt, 2015). However, from considering the feature list, it can be inferred that the selection of a proper subset of these meta-features mainly follows the expertise of data analysts and is mostly subjective and non-systematic. Fulcher (2017), Fulcher and Jones (2014), Timmer et al. (1993) indicated that it is difficult to conclude whether new features presented by researchers are better than existing alternatives. Moreover, it is hard to determine whether methodologically complicated meta-features outperform simpler alternatives. Fulcher et al. (2013) discussed that a threshold on the simple standard deviation computed for each time series provides comparable classification performance on different problems, undermining the need for computing nonlinear features or the use of complex classification algorithms. Commonly, the main criteria for selection of meta-features is overall predictive accuracy using the subset of meta-features. However, based on Giraud-Carrier (1998), other criteria such as computational complexity, expressiveness, compactness, and prior knowledge encoding may have equal importance.

Early efforts of meta-learning studies, which include meta-features in forecasting, have been summarized in Smith-Miles (2009). The number of implemented meta-features in papers is different, as while some researchers incorporated small sets of 6-13 features (Prudêncio and Ludermir, 2004, Wang et al., 2009, Venkatachalam and Sohl, 1999), others applied larger sets of 25- 38 features (Lemke and Gabrys, 2010a, Shah, 1997, Meade, 2000, Lemke and Gabrys, 2010b, Kück et al., 2016b).

Until now, three main classes of meta-features have been suggested: 1. statistical and information-theoretic characterization, 2. model-based features, and 3. “landmarkers” (Brazdil et al., 2008). The first group estimated the statistical features of the dataset. For example, standard deviation of series, skewness, kurtosis, length of series, entropy of series, and number of exogenous variables are some of these features. As an example of the second type, one can build a decision tree from a dataset and capture the properties of the tree such as maximum tree depth, shape, and tree imbalance as model-based features. Finally, the last class of so-called landmarkers exploits information obtained from the performance of a set of learners/forecasts and their accuracy is used to characterize the time series. The differences between the model-based features and landmarkers are related to the fact that the latter does not come under model characteristics, but performance measures of the built model. Kück et al. (2016b) used the errors on the training, validation, and test set of the neural network as landmarkers.

Characteristics of a learning algorithm with meta-features and gaining a better view about the problem performance is a narrow research topic with limited literature on it (Vanschoren and Blockeel, 2006). For example, Hyndman et al. (2015b) used 18 meta-features to characterize the time series, then applied a two-dimensional principal component decomposition to the features to detect unusual series with bivariate outlier detection methods. Kang et al. (2017) visualized 6 meta-features into a two-dimensional principal component feature space and explained which algorithm can be suited to forecast a specific type of time series in different parts of the space. Talagala et al. (2018a) used a set of 33 meta-features including different variant of Autocorrelation function (ACF) as well as some ETS model base meta-features. Montero-Manso et al. (2020) applied 42 meta-features mainly from Talagala et al. (2018a) study as well as model based meta-features from ARCH and GARCH.

A summary of meta-features used in the literature is presented in Table 2.3-1. Hyndman, et al. (2015) and Kang et al. (2017) studies are not in the context of meta-learning; however, they use meta-features for time series evaluation. Table 2.3-1 classifies the meta-features into four sections which are statistical and information-theoretic features, model-based features, statistical tests, and land markers respectively. It can be inferred that land markers are hardly used since the learning models have to be applied on the whole data set, thus not benefitting from the speed of filter meta-features. Moreover, establishing statistical tests, a common way for time series trend and seasonality detection, is largely overlooked in this area. Statistical tests are a “filter” approach which only evaluate the time series and label them without learning from the past. It is truly rational that using statistical test for detecting the most important attributes of time series (level, trend, and seasonality) are more meaningful, than many statistical features such as skewness, lumpiness, etc. Statistical tests for determination of trend and seasonality of time series are commonly used between researchers (Kendall and Ord, 1990, Hamed, 2008, Sun and Fang, 2017, Sayemuzzaman and Jha, 2014), proposing better forecasting algorithms along with characterising the time series. Specifically, when the presence of seasonal or trend patterns are not entirely visible, these tests are more useful (Kourentzes, 2017). Therefore, for the first time, we establish different combinations of statistical tests in meta-learning to clarify the time series by determining the statistical relationships in the data, e.g., stationarity, seasonality, trends, and nonlinearity.

Table 2.3-1. Summary of implemented meta-features in different studies

Features	Arinze (1994)	Prudêncio and Ludermir (2004)	Wang et al. (2009)	Lemke and Gabrys (2010a)	Matijaš et al. (2013)	Kück et al. (2016b)	Hyndman et al. (2015b)	Kang et al. (2017)	Talagala, et al. (2018)	Montero-Manso, et al. (2020)	Ma, and Fildes (2021
Standard deviation		✓		✓	✓		✓				
Mean					✓		✓				
Minimum					✓						
Skewness		✓	✓	✓	✓	✓					
Kurtosis		✓	✓	✓	✓	✓					
Length		✓		✓	✓				✓	✓	
Granularity					✓						
Periodicity (per)	✓		✓		✓	✓					
Traversity					✓						
Trend	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Seasonal period								✓			
Entropy							✓	✓	✓	✓	✓
Step changes				✓							
stability											✓
hurst											✓
Turning points	✓	✓		✓							
Kullback-Leibler (kl) score							✓				
index of the maximum KL score							✓				
Curvature							✓		✓	✓	✓
Peak							✓		✓	✓	
Seasonality			✓	✓		✓	✓	✓	✓	✓	
Linearity							✓		✓	✓	✓
Lumpiness							✓		✓	✓	✓
Trough							✓		✓	✓	
flat spots							✓		✓	✓	✓
Level shift							✓		✓	✓	
Number of crossing points							✓		✓	✓	
Spikiness							✓		✓	✓	✓
Variance change							✓		✓	✓	
Fickleness					✓						
Self-similarity			✓			✓					
Serial correlation	✓	✓	✓			✓					
Model-based features	✓	✓							✓	✓	
Predictability measure				✓							
Nonlinearity measure			✓	✓		✓					✓
Largest Lyapunov exponent			✓	✓		✓				✓	
Durbin–Watson				✓							
Exogenous					✓						

Features	Arinze (1994)	Prudêncio and Ludermir (2004)	Wang et al. (2009)	Lemke and Gabrys (2010a)	Matijaš et al. (2013)	Kück et al. (2016b)	Hyndman et al. (2015b)	Kang et al. (2017)	Talagala, et al. (2018)	Montero-Manso, et al. (2020)	Ma, and Fildes (2021
Highest Acf					✓				✓	✓	
Acf / diff Acf							✓	✓	✓	✓	✓
Pacf / diff Pacf				✓							✓
Domain frequency		✓	✓	✓	✓	✓					
Diversity features				✓							
Optimal Box–Cox transf. par.								✓			
Mean of five first autocorrelations		✓					✓		✓	✓	✓
Test of Turning Points		✓					✓		✓	✓	✓
Test of autocorrelations		✓					✓		✓	✓	✓
Alpha											✓
Beta											✓
ARCH.LM											✓
Recurrence quantification analysis (RQA)						✓					
Mann-Kendall						✓					
Spearman's Rho						✓					
Cox- Stuart						✓					
Chi-square test						✓					
Kruskal-Wallis test						✓					
Train Error						✓					
Validation error						✓					
Train error ranking						✓					
Validation error ranking						✓					

By analysing the features presented in Table 2-2, we can find that still there is a lack of research in using the data driven features extracted from more advanced methods like unsupervised deep networks. We consider these group of features in the next chapters.

2.4 Experimental Design

2.4.1 Methodology

Our aim is to assess the empirical accuracy of the meta-learning approach in forecasting model selection, in comparison to established benchmark approaches. We train multiple classification algorithms to predict the most probable class membership of time series based only on the input features of statistical trend and seasonality tests, in an attempt to select the right base learner to predict future realisations of time series.

In order to limit complexity, we constrain our classification to the four univariate classes of constant, trend, seasonal, and trend-seasonal time series of the Pegels-Gardner classification (Pegels, 1969, Gardner Jr, 1985). These four archetypical patterns represent the biggest theoretical differences in data generating processes of observed time series, each requiring a specific model form to be adequately captured and extrapolated. Should a meta-learner, or indeed any other benchmarking approach of model selection, fails to make these fundamental distinctions, it should result in increased forecast errors. Moreover, specific type of patterns like damp seasonal or damp trend are not considered here.

To match the classes of the data generating process with base learners, both for meta-learning and alternative model selection benchmarks, we consider four Exponential Smoothing (ETS) methods, which are well-established in commercial forecasting software and research packages, and thus enhance the relevance of our experimental findings for researchers and practitioners (Gardner Jr, 2006). A survey of forecasting practices identified exponential smoothing families as the most frequently used methods (Weller and Crone, 2012), with a proven track record in practice (Gardner Jr, 2006) and a proven relative performance compared to more complex methods (Makridakis and Hibon, 2000, Armstrong, 2006, Crone et al., 2011).

The task of the meta learner is thus to select the Exponential Smoothing algorithm from the set of four archetypical algorithms of a constant model ETS(ANN), a seasonal model ETS(ANA), a trend model ETS(AAN), and a trend seasonal ETS(AAA). Each chosen algorithm is parameterised using maximum likelihood on the training data using the Smooth package in R (Svetunkov 2017).

Predictive classification accuracy for a given multi-class classification problem depend on the type of classification algorithm used, the input features utilised, and the data conditions. Consequently, our experimental design seeks to assess the sensitivity of the results across:

- different classification algorithms in multi-class classification, including Decision Trees (DT), Neural Networks (NN), Support Vector Machines (SVM) and popular ensemble extensions such as Random Forests, Bagging, and Stochastic Gradient Boosting,
- different sets of input features, limited to statistical tests of trend and seasonality and extended to include features used in previous meta learning studies,
- different splits of the given dataset in training, validation and test data of 60:40, 70:30, and 80:20 to influence the ability for the classifier to learn the classification task on training data, and to generalise it to unseen test data. It should be noted that the results may alter with the change in the size of the training/test dataset because of shifting in data distribution.

Therefore, in the first experiment, we would like to explicitly investigate whether the statistical tests are capable of detecting the level, trend, seasonality, and trend-seasonality in different time series.

Secondly, to evaluate the potential of statistical tests as meta-features, we compare their forecast performance versus the statistical, information-theoretic, and model-based features implemented in Hyndman, Wang, and Laptev (2015a). Their features later presented as TsFeature and including mean, variance, the first order of autocorrelation, trend, linearity, curvature, seasonality, peak, trough, spectral entropy, lumpiness (changing variance in the remainder), spikiness, level shift (using rolling window), variance change, flat spots (using discretization), number of crossing points, Kullback-Leibler (kl) score, and an index of the maximum KL score.

Finally, in the third experiment, we seek to assess the robustness of our approach by considering different training and test ratios consisting of 80%-20%, 70%-30%, and 60%-40% to enhance the reliability of the experiment. Different sample size of training data may result in different features selected, different parameters estimated and thus different relative ranking of model selection approaches, assessing the robustness of our proposed approach on different model forms and parameters. We select a specific random number to avoid different permutations in the cross-validation.

The suggested approach has minimum assumptions or requirements and is easy to apply as a model selection approach. However, in this chapter, we investigate the ability of meta-learning to capture the trend and seasonality of time series and therefore consider four exponential smoothing variants that have clear patterns of trend and seasonality. Here, our main aim is pattern detection with meta-learning and not solely increasing the forecasting performance. However, for the forecasting performance, you would need more base forecasters.

The practical importance of the chapter is that it enhances the accuracy of model selection further, while decreasing the selection time of commonly used wrapper model selection. In the big data era, which we are encompassed with lots of time series, the proposed model can get a reasonable accuracy within dramatically lower time. It is worth mentioning that because of the pattern detection spirit of this chapter, we solely focus on selection not combination of base forecasters.

It may happen that different selection criteria lead to different forecasts which makes uncertainty in identifying the best model; however, this problem can be alleviated using meta learning. Meta learning uses the character of time series to select the relevant forecast which result in the best accuracy in the similar data driven time series. This does not necessarily need to determine one best forecast for all-time series.

2.4.2 Meta-learners for classification

At the core of meta-learning, the choice of meta-learners may impact the classification performance and thus its forecasting performance. In our experiments the meta-learner is tasked with learning a relationship between input features of statistical tests computed on time series, and the final performance of four archetypical base learners of ETS each representing a class of constant, trended, seasonal or trend-seasonal time series. As different classification algorithms partition feature space differently, we seek to evaluate a selection of different meta-learners in order assess similarities or differences in their performance on the same input features, most notably Decision Trees, Neural Networks and Support Vector machines, as well as ensembles of Decision Trees in the form of Bagged Trees, Random Forrest and XGBoost.

- Artificial Neural Networks:

Artificial neural networks are well-established algorithms, which are inspired by the functioning of biological nervous systems, and capable of multiclass-classification (Priddy and Keller, 2005). In our application as meta-learners, the inputs of a neural network are vectors of features corresponding to the statistical tests, which are weighted and combined by linear filters to become inputs of hidden layers using non-linear combinations. In this study, feedforward neural network architecture of a multilayer perception (MLP) is used, with sigmoid activation function. For the number of hidden layer nodes, we use a convenient rule which suggests the mean value of input variables and output variables. So, we use two hidden layers that first one has 17 nodes and the second one has 7 nodes. The number of epochs is set to 200. We also train the network with different hyper parameters and find the mentioned parameters as the most optimised specification in the training set. Since these hyper parameters are determined in the training set, we control our results from overfitting.

- Decision Tree:

The Decision Tree (DT) algorithm is a non-parametric and non-linear machine learning technique. This technique takes advantage of a hierarchical structure for recursively segmenting training data and therefore it has a great flexibility and interpretability in data analysis. In this chapter, we are using CART (classification and regression tree) and the evaluation function used for splitting in classification trees is the Gini index. The Gini index can be stated as: $Gini(t) = 1 - \sum[P(K|t)]^2$ where $P(K|t)$ is the proportion of finding the data class K in node t (node purity). The aim is to minimize the Gini index. From the formula it can be inferred that if the classification is done in a perfect way, the Gini index would be zero (Friedman et al., 2001). For implementing this method, we use ‘rpart’ package in R (Therneau et al., 2018) with 10-fold cross-validation and a maximum depth of 10 for the tree.

- Support Vector Machine:

Support Vector Machine (SVM) are an algorithm, where for a set of data $(x_1, y_1), (x_2, y_2) \dots, (x_n, y_n)$ where $(x_i, y_i) \in R^2$ are the respective input and output with the help of kernel functions the input data is converted to a new higher dimensional space which is called the feature space. In the feature space an estimated function such as: $g(x) = \alpha x + \beta$ will be considered, which is in fact the equation for a hyper plane in the feature space. Finally, with the help of Lagrange multipliers the equation for the hyper plane can be rewritten as: $g(x, \beta_i, \beta_i^*) = \sum_{i=1}^{n}(\beta_i - \beta_i^*)K(x, x_i) + c$ where $K(x, x_i)$ is the kernel function. More details can be found in Vapnik (2013). In this chapter, we use ‘SVM’ function from ‘e1071’ package in R (Dimitriadou et al., 2006) with radial kernel function.

- Random Forest:

Random Forest (RF) is an ensemble classifier, made by a combination of decision trees which are created by recursive partitioning. In order to construct the RF model, with the help of bootstrapping, new sets of training data are created and then RF randomly chooses the variable for each set for better diversity in results. Then RF starts creating decision trees for each group with respective variables. Finally the outcome forest of trees will be combined and the average of the predictions will be considered as the result (Friedman et al., 2001). For testing the accuracy of the RF, Out-of-Bag (OOB) data which are the samples that were not selected in bootstrapping in the RF procedure, can be used. Choosing randomly a subset of predicting variables help the model to have less computation and to avoid the over-fitting problem. For implementing this method, we use Caret package in R with “ntree=500”, which indicates the number of trees that should be grown with 10-fold cross-validation.

Bagging, which is an abbreviation for Bootstrap aggregation, is an ensemble technique that uses a bootstrap to generate samples of the original data. In prediction problems this algorithm averages the prediction over a collection of bootstrap samples and the class of a new observation is the most selected class among the number of trees constructed on the bootstrap samples (Breiman, 1996). In RF, trees are grown deep without pruning. But by building sufficient trees, the over-fitting is less probable to happen. The algorithm can be described by a loop where for I = 1: B we generate a bootstrapped sample of the data, then create unpruned decision trees on the samples, and then average over all the outcomes, and end. In this method, like the case of random forests before, OOB data can be used to evaluate the performance of the model (Kuhn and Johnson). In this chapter, we use the bagging function from 'ipred' library (Peters et al., 2009), with 'nbag=25' as the tuning parameter which indicates the number of bootstrap replications. Also, we use 10-fold cross validation as training control.

- Extreme Gradient Boosting:

The Extreme Gradient Boosting (XG-boost) is an optimized implementation of boosting method (Chen and Guestrin, 2016). In Additive learning of XGBoost, the first learner is fitted on the whole data, and the next learners are fitted to the residuals of the former learners. In fact, each learner is fitted using information from previously fitted learners. The general function for the prediction at each step is presented as follows:

$$\widehat{f_j^t} = \sum_{i=1}^{t} f^i(x_j) = \widehat{f_j^{t-1}} + f^t(x_j) \tag{2-8}$$

where $f^i(x_j)$ is the learner at step i, $\hat{f}_j^t$ is prediction at step t, and x_j is the input variable. Unlike the random forest and bagging, gradient boosting methods are prone to over-fitting if the number of trees is too large. A computation procedure for preventing over-fitting can be found in Fan et al. (2018). We implement this model using the '*xgboost'* package in R (Chen et al., 2015) with 'nround=100' that indicates the maximum number of iterations and for this number we set the learning rate to 0.2 (eta=0.2). It is worth noting that all the hyper parameters tuning are implemented in the training set for all the classifiers and therefore we avoid trapping in the overfitting.

2.4.3 Feature encoding

All classifiers have in common that their output corresponds to a multi-class classification problem of four classes, providing either the true class membership or the probability of belonging to one class of single ETS(ANN), Trend ETS(AAN), Seasonal ETS(ANA), or Trend-Seasonal ETS(AAA). The true class here means the algorithm that has the lowest error.

All meta-learners receive the same meta-features to learn the mapping of input features to true class membership. These features include statistical tests of Mann-Kendall trend test, ADF test with alternative hypothesis no Constant - no Trend, Constant- no Trend, and Constant – Trend, Spearman's rho (SR) trend test, Chi square seasonality test, Cox-Stuart trend test, Cox-Stuart dispersion test, Friedman seasonality test, Pearson correlation multiplicative seasonality test, ACF for detecting seasonality, trend, and level, Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests, Kruskal-Wallis test of seasonality, and linear coefficient trend test. All tests are provided as a metrically scaled variable of the corresponding p-value of the statistical tests [0.00, …, 1.00], in addition to the ACF and ADF tests which are presented in four states with critical values of $p<0.01$ corresponding to strong significance of the test, $0.01<p<0.05$ as moderate significance, $0.05< p<0.10$ as weak and $p>0.1$ as not significant. Note however that the critical level of significance is not information provided, so must be learned by the meta-learner. The list of meta-features is presented in Table 2-3.

It should be noted that the list of features presented in Table 2-3 is not a subset of Table 2-2. Rather, here we focus only on statistical tests, and we want to measure the performance of meta-learner by using these features. Table 2-2 includes the predefined features that have been used in recent studies, but it does not include all the statistical tests that are in Table 2-3. Also, Table 2-3 can be extended, and more statistical tests can be added to it, but we considered those statistical tests that have been used more in recent studies.

Table 2.4-1. List of meta-features

Number	Features name	Description
1	Mann-Kendall	P-value for the Mann-Kendall test
2	SpearmanRho	P-value for the SpearmanRho test
3	Cox-StuartTrend	P-value for testing trend
4	Cox Stuart dispersion	P-value for testing dispersion trend
5	LinearCoefficient	P-value for the coefficient test of linearity
6	ChiSqMod	P-value for the Chi Square seasonality test
7	Kruskal-Wallis	P-value for the Kruskal-Wallis test
8	F-test	P-value for the Friedman test 2
9	Friedman test	P-value for the Friedman test
10	Kpss	P-value for the KPSS test
	Multiplicpval	P-value for the Pearson test of multiplicative seasonality
11	adf-R	P-value for the ADF test
12	ADF (no constant-no trend)	In four states including No, Weak, Moderate, Strong
13	ADF (constant-no trend)	In four states including No, Weak, Moderate, Strong
14	ADF (constant-trend)	In four state including No, Weak, Moderate, Strong
15	ACF-Stationary	In four states including No, Weak, Moderate, Strong
16	ACF-Trend	In four states including No, Weak, Moderate, Strong
17	ACF-Seasonality	In four states including No, Weak, Moderate, Strong

2.4.4 Forecasting models as base learners

One of the core components of meta-learning is the base learner, the forecasting algorithms to be selected by the classification algorithm. In this research, exponential smoothing (ETS) forecasting methods based on Gardner’s classification (Gardner Jr, 1985) with a fully automatic methodology using state space models developed by Hyndman et al. (2002) is used. All Gardner’s methods can be summarized by the following equations:

$$Y_t = h(x_{t-1}) + k(x_{t-1})\varepsilon_t \qquad (2\text{-}9)$$

$$x_t = f(x_{t-1}) + g(x_{t-1})\varepsilon_t \qquad (2\text{-}10)$$

where $x_t = (l_t, b_t, s_t, s_{t-1}, ..., s_{t-(m-1)})$ is a state vector, and $\{\varepsilon_t\}$ is a Gaussian white noise process with mean zero, variance σ^2 and $\hat{Y}_t = h\ (x_{t-1})$ is the one-step-ahead forecast. In this chapter, we only applied four ETS models which are: single exponential smoothing (SES), trend ETS (Holt model), seasonal ETS (Winter model), and trend-seasonal ETS (Holt-Winter model). Since we are using the Forecast package in R (Hyndman and Khandakar, 2007), these models are denoted as ANN, AAN, ANA, and AAA respectively.

Other alternatives exist instead of ETS as base learner, such as artificial neural networks or other machine learning methods; however, they provide limited or no insights into how the forecasts are produced and which data properties are considered in the forecast (Sagaert et al., 2018), whilst with ETS the explicit model form determines their suitability to a data generating process, e.g. selecting a seasonal ETS(ANA) model for a seasonal time series without trend.

2.4.5 Benchmark dataset

As a dataset, we employ the reference benchmark dataset of the NN3 competition. Note that the four classes of time series patterns capture all differences of the data generating processes in the benchmark dataset from the NN3 competition (Crone et al., 2011). The empirical benchmark dataset of the NN3 competition contains 111 monthly time series of industry sales, originally derived from the popular M3 competition of which most are from the economics and business sectors (Crone et al., 2011). Of the 111-time series, 46 are short (<60 data points), while the rest are considered long (80 to 126 data points), which allows an assessment of the accuracy across different data conditions of time series length, a characteristic which may influence the precision of the statistical tests as well as the representativeness of empirical forecast errors and information theoretic metrics to be estimated.

It should be noted that the length of time series may affect the quality of extracted meta-features. Therefore, in chapter 4 we analysed the performance of meta-learning on different length of the series. By the way, in this study, we did not split the time series into different fragments.

2.4.6 Forecasting model selection benchmarks

We seek to assess the efficacy of meta-learning for model selection against established model selection benchmarks, which include different wrapper and filter approaches identified in our literature review.

To begin, we consider the traditional form of aggregate model selection of applying each base learner ETS(ANN), ETS(ANA), ETS(AAN) and ETS(AAA) to all time series within the dataset, following the traditional suggestions by Fildes (1989). Aggregate model selection by terminology presumes that no (individual) model selection per times series implicitly equals a model selection for the dataset, so qualifies as a filter approach, normally decided by a human expert for reasons of convenience or limited computational powers. It still serves as a valid benchmark, most notably in the form of aggregate model selection of the Naïve and the Seasonal Naïve benchmark methods.

With the growing compute power, both forecasting research and practice turned to employ model selection based on wrapper-based approaches, utilising either empirical performance measures of forecast accuracy such as sMAPE or MASE, estimated in-sample or out-of-sample with fixed or rolling origins, or alternatively on information criteria such as AIC, AICc or BIC. Considering the properties of the NN3 dataset, including short time series, we employ the AICc with correction for small sample sizes with the benchmark results denotes selAIC (Burnham and Anderson, 2003). Since all our base learners are exclusively from the same model family of ETS, a use of IC seems permissible.

As a further benchmark we compare accuracy against individual model selection of ETS(ZZZ) automation developed to detect the appropriate exponential smoothing models by Hyndman and Khandakar (2007). This procedure selects the best fitting ETS candidate from different exponential smoothing models using the AIC criterion. Minimizing the AIC is asymptotically equivalent to minimizing the one-step-ahead out-of-sample MSE; a smaller AIC means better forecasts, and ZZZ can determine the optimized model of time-series based on minimizing the AIC. Therefore, as a further benchmark we add selETS(ZZZ) utilising a wrapper with AIC selection.

Although filter approaches using statistical tests are rarely used in practice, we consider the evaluation of their accuracy as an important intermediate step to meta-learning using the said tests. Consequently, we include statistical tests for seasonality and trend as benchmark. We run each statistical test independently, e.g., Cox-Stuart for detecting trend and the Friedman-test for detecting seasonality and combine both for a one-in-four class membership. As multiple test combinations are feasible, we constrain our analysis to four popular combinations of two tests.

Finally, to determine a lower bound of forecast accuracy, we identify the lowest achievable error by selecting for each series that base learner with the lowest test error ex post, i.e., the error resulting from a perfect selection, denoted as $selMinError$.

To summarise, we compare meta-learning model selection with representatives of all other approaches to forecast model selection, in order to assess the relative capability of the meta-modelling approach in a comprehensive design of multiple benchmark approaches.

2.4.7 Assessing predictive accuracy

Predictive accuracy may be assessed in two ways: first, as meta-learning is foremost a multiclass classification problem, through the classification accuracy of the meta-learner; second, we may assess the accuracy of meta-learning through the final forecasting accuracy estimated from the base-learner selected by meta-learning for each of the time series.

First, we consider the simple average classification accuracy of each algorithm’s multiclass prediction in learning and generalising the meta-learning task, i.e.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (2\text{-}11)$$

as the percentage of predicted true positives (*TP*) and true negatives (*TN*) over all predicted instances of *TP*, *TN*, false positives (*FP*), and false negatives (*FN*). This simple metric seems permissible as the NN3 dataset is balanced with regard to seasonality and short vs long series (so no severe or class imbalances need to be considered which would warrant more complex metrics such as multiclass ROC-curves and AUC, see, e.g. Fawcett (2006). The true class membership of each time series is determined ex-ante according to its identified time series pattern c_p= [*constant*; *trend*; *seasonal*; *trend-season*] from the ETS forecasting algorithm with the lowest sMAPE test error. Note that this may induce potential misclassification in a few cases, e.g., by selecting a seasonal pattern as the true class for a time series due to the lower ETS(ANA) test-error on the last 12 months of test error, which has been caused by a diminished local time trend despite the global time series pattern resembling a trend-seasonal pattern.

In addition to accuracy in predicting class membership of the meta-learner, we assess the forecasting accuracy of the chosen base-learner applied to the forecasting task. Once an ETS algorithm is chosen, we assess a rolling origin symmetric mean absolute percentage error (SMAPE) from the actuals y_t and corresponding forecast $\hat{y}_t$ for each forecasting horizon h= 1, …, *H* and across multiple forecasting origins *i*= 1, …, *i*+*I*-1 for each time series:

$$Rolling\,Origion\ sMAPE^{h}_{i,i+I-1}(y,\hat{y}) = \frac{1}{I}\sum_{k=i}^{i+I-1} sMAPE^{h}_{i}(y,\hat{y}) \tag{2-12}$$

with

$$sMAPE^{h}_{i} = \frac{200}{h}\sum_{t=i+1}^{i+h}\frac{|y_t-\hat{y}_t|}{|y_t|+|\hat{y}_t|} \tag{2-13}$$

In order to make a better comparison, we define a metric called Relative sMAPE (RelsMAPE) for measuring forecasting accuracy. It is the ratio of the sMAPE between the candidate model and the benchmark model (aggETS(AAA)).

$$RelsMAPE = \frac{sMAPE_i^h}{sMAPE_{\text{aggETS(AAA)}}^h} \tag{2-14}$$

Thus, the empirical evaluation employs a fixed horizon time window *H*=12, multi-origin *I*=6, out-of-sample evaluation on each time series, withholding 18 observations for testing in order to ensure more valid and reliable estimates of forecast accuracy than feasible form a single holdout-validation window (Tashman, 2000). These errors are then averaged across all time series in the dataset. Given the test set of size *m* and horizon *h*, for each data set *q* = *m*-*h*+1 the ETS parameters are reoptimized.

2.5 Experimental results

2.5.1 Meta-learning versus alternative model selection approaches

First, we seek to assess the overall accuracy of meta-learning using statistical tests for time series patterns. Table 2-4 presents for each algorithm the forecast accuracy measured in RelsMAPE, and the classification accuracy on training and test data (80-20% for train-test percentage), including the respective ranks of algorithms. Table 2.5-1 also reports the RelsMAPE of aggregate and individual selection benchmarks, in addition to the naïve and seasonal naïve for the train and test, corresponding with ranks of them.

Table 2.5-1. Classification and Forecasting Results of Model Selection Approaches (80%-20% holdout)

	RelsMAPE		Accuracy		Rank RelsMAPE		Rank Accuracy	
	Train	Test	Train	Test	Train	Test	Train	Test
selMinError	0.843	0.903	100.000	100.000	-	-	-	-
*meta*DecisionTree	0.898	0.959	56.000	36.600	5	4	7	12
*meta*NN	0.880	0.959	100.000	46.670	4	4	1	6
*meta*SVM	0.916	0.979	76.000	53.330	6	7	5	4
*meta*RandomForrest	0.843	0.952	100.000	60.000	1	3	1	2
*meta*BagTree	0.843	0.945	100.000	63.330	1	1	1	1
*meta*XGBoos*t*	0.843	0.945	100.000	56.660	1	1	1	3
*sel*AIC	0.976	0.979	54.000	53.330	10	7	8	4

	RelsMAPE		Accuracy		Rank RelsMAPE		Rank Accuracy	
	Train	Test	Train	Test	Train	Test	Train	Test
*sel*ETS(ZZZ)	0.946	1.028	57.000	44.000	7	12	6	8
*agg*ETS(ANN)	1.108	1.234	10.000	14.000	16	15	18	17
*agg*ETS(AAN)	1.187	1.241	14.000	12.000	17	16	17	18
*agg*ETS(ANA)	0.946	1.034	40.000	38.000	7	13	11	11
*agg*ETS(AAA)	1.000	1.000	35.000	36.000	13	10	13	13
Naive	1.373	1.497	24.000	35.000	18	18	16	14
Seasonal Naive	1.012	1.386	31.000	22.000	14	17	15	16
test CS & Fr	0.988	0.986	51.000	40.000	12	9	9	9
test MK & Fr	0.976	1.007	45.000	30.000	10	11	10	15
test CS& Sp	1.030	1.103	40.000	40.000	15	14	11	9
*test*CS & KW	0.964	0.966	34.000	46.670	9	6	14	6
mean of meta	0.873	0.959	88.667	52.765	3.00	3.33	2.67	4.67
mean of sel	0.964	1.007	55.500	48.665	8.50	9.50	7.00	6.00
mean of agg	1.060	1.131	24.750	25.000	13.25	13.50	14.75	14.75
mean of test	0.988	1.014	42.500	39.168	11.50	10.00	11.00	9.75

Overall, meta learning classifiers with statistical tests of time series patterns as input features outperforms all other approaches of forecasting models selection, including aggregate model selection, individual model selection using AIC (selAIC and selETS), and simple combinations of statistical tests on both classification accuracy and forecast errors. To generalise the indication of individual errors for groups of algorithms, we compare average ranks across groups of algorithms at the bottom of the Table 2-4: the group of metaTest shows an average rank of 3.33 with an average RelsMAPE of 0.959 while individual selection ranks 9.50 on average with RelsMAPE of 1.007, and tailed by aggregate selection with 13.50 average rank and RelsMAPE 1.131.

With regard to individual algorithms, the ensemble meta learners metaBagTree achieve the lowest forecast error of 0.945 RelsMAPE and the highest classification accuracy of 63.33%, followed by the ensemble of metaXGBoost and metaRandomForrest with 0.945 and 0.952 RelsMAPE and 56.66% and 60.00% classification accuracy respectively. The three ensembles of meta-learners are followed by individual meta-learners of metaNN, metaDecisionTree, and metaSVM in error and accuracy, with only metaSVM showing inferior performance to alternative model selection approaches on the test data. It is noteworthy, that all three ensembles of meta-learners lead the underlying single decision tree algorithm on both metrics, confirming findings from the literature that combinations of classifiers outperform individual ones. Extended experiments could consider ensembles of NNs and SVMS, although no direct equivalents of Random Forests or XGBoosting algorithms exist for other base learners. Overall, we conclude that all meta ensembles using statistical tests outperform individual meta learners of the same feature set, followed by all other approaches.

To aid interpretation of the algorithms' performance we determine a lower bound of forecast accuracy *selMinError* as the lowest achievable error given all base learners selected for each series, i.e., the error resulting from a perfect selection, resulting in a bound of 0.903 test RelsMAPE. Meta-learners using statistical tests for times series patterns get closest to this bound, with errors of 0.945, 0.952 and 0.959 RelsMAPE being the lowest of all algorithms.

Aggregate selection of a single ETS algorithm across all-time series shows inferior forecast accuracy to model selection, as expected following the recent research. The least damaging aggregate selection strategy would employ a trend seasonal ETS(AAA) for all series, followed by a seasonal model ETS(ANA). This is intuitive given the balanced nature of the NN3 dataset: the aggregate seasonal algorithms would be capable of correctly capturing the 50% of seasonal time series well but would underperform on the other 50% of non-seasonal series where the algorithms would still estimate seasonal coefficients, thus extrapolating noise into the future.

Surprisingly, simple combinations of statistical test for seasonality and trend, which are rarely used in statistical forecasting perform on a par with individual model selection using AIC (selAIC) or AIC in ETS selection *selETS*(ZZZ), which are considered the current standard in forecasting model selection, and clearly outperform aggregate model selection. Most notably, in combining Cox-Stuart with Kruskal-Wallis test achieves a 0.966 RelsMAPE and 46.67 accuracy; note though that the result may be biased as the selection of statistical tests listed in Table 2-4 shows only the most promising combinations of trend and seasonality tests by accuracy: many inferior combinations are excluded for readability. In comparison to meta-learning though, the leading combinations of statistical tests show both an inferior classification accuracy as well as forecasting accuracy, indicating that a more complex combination of all tests yields improved accuracy. The reason that the combination of statistical tests has unfavourable results is that they cannot detect patterns. Therefore, they do not have acceptable results when they are used as meta-learners.

Overall, we observe that classification accuracy and forecast accuracy are correlated, which indicates that an improvement in classification induces an improvement in forecast error. This is an intuitive yet important finding, as it provides anecdotal evidence as to the efficacy of using classification for forecasting model selection, and the underlying narrative that selecting an adequate model class matching the time series pattern improves accuracy.

Furthermore, it should be noted that the run-time of the competing approaches varies significantly: while meta-learning runs only a single ETS model (predicted to be the best given the statistical tests), individual model selection needs to compute all base learners, assess accuracy, and then select. Consequently, as long as statistical tests are executed faster than an ETS model, meta-learning promises increased efficiency on top of increased accuracy.

To better understand the performance of meta-learners in the data set, comparing their results with the true class performance is necessary. As mentioned in the previous sections, true class is the class for a time series with the least error. We depict the relative error distribution diagram [see Figure 2-2] to better understand the results. Our criterion for identifying the best model is the AIC measure. To draw the relative error distribution diagram, we define the relative error as a ratio, the numerator of which is the error related to the class predicted by the meta-learner, and the denominator is the error related to the true class. Since the true class always has the least error, this ratio is always greater than 1 (the more this ratio deviates from 1, the poor performance of meta-learners for that time series). Figure 2-2 shows two meta-learners' relative error distribution diagram, RF and Xgboost, for the training and test data of 80%-20% split.

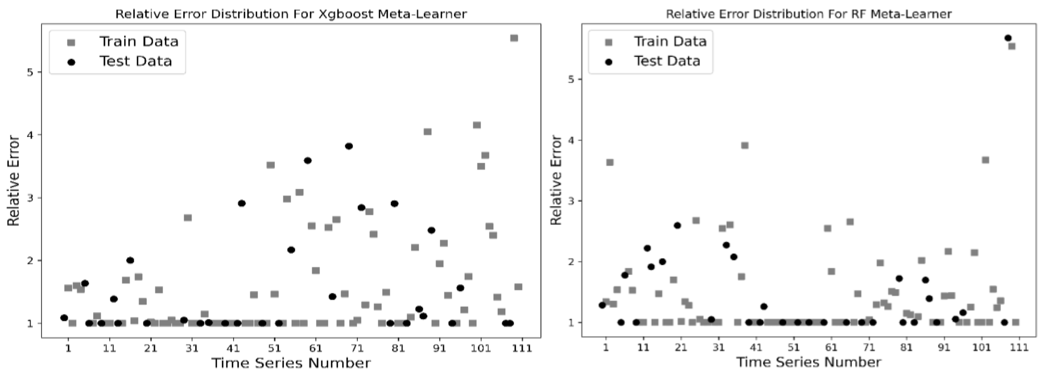


Figure 2-2. Relative error distribution for RF and Xgboost

According to Figure 2-2, it is clear that RF has performed better than Xgboost, which is consistent with the results obtained in Table 2-4.

2.5.2 Meta-learning using "time series tests" versus alternative features

We seek to assess the accuracy of meta-learning using statistical tests for time series patterns in comparison to alternative meta-features used successfully in other prominent stories. Table 2.5-2 presents for each algorithm the forecast accuracy measured in sMAPE, and the classification accuracy on training and test data, including the respective ranks of algorithms, for the new meta-features based on statistical tests versus those in Hyndman et al. (2015b) as a baseline features.

Table 2.5-2. Classification and Forecasting Results of Different Meta Features (80%-20% train test holdout)

	Statistical Test Meta-Features				Hyndman, et al. (2015) Meta-Features			
	sMAPE %		Accuracy %		sMAPE %		Accuracy %	
	Train	Test	Train	Test	Train	Test	Train	Test
metaDecsionTree	14.9	13.9	56.00	36.60	15	**13.75**	55.50	43.33
meta NN	14.6	13.9	100.00	46.67	15.1	14.52	98.00	37.00
metaSVM	15.2	14.2	76.00	53.33	14.5	14.35	70.37	40.00

metaRandomForrest	14	13.8	100.00	60.00	13.97	14.06	100.00	46.67
metaBagTree	14	**13.7**	100.00	63.33	13.97	14.35	100.00	40.00
metaXGBoost	14	**13.7**	100.00	56.66	13.97	14.09	100.00	43.30
Mean Values	14.45	13.87	88.66	52.77	14.28	14.12	85.17	42.66

	Rank sMAPE %		Rank Accuracy %		Rank sMAPE %		Rank Accuracy %	
	Train	Test	Train	Test	Train	Test	Train	Test
metaDecsionTree	5	4	7	12	5	1	7	5
meta NN	4	4	1	6	6	10	4	12
metaSVM	6	7	5	4	4	7	5	7
metaRandomForrest	1	3	1	2	1	3	1	2
metaBagTree	1	1	1	1	1	7	1	7
metaXGBoost	1	1	1	3	1	4	1	6
Mean Ranks	3	3.33	2.67	4.67	3.00	5.33	3.17	6.50

We compare the performance of all meta learners, single and ensembles, using otherwise identical settings on two different meta-features: 17 statistical tests of time series patterns, and 21 meta-features used in Hyndman et al. (2015b).

Both meta-learners are built using different meta-features of statistical tests versus those of Hyndman, et al. perform well, showing comparatively low average forecast errors on test data of 13.87% and 14.12% and classification accuracies of 52.77 and 42.66 respectively, indicating their capability of separating classes and predicting suitable base learners. Overall, meta-learners using statistical tests outperform those of Hyndman, et al. showing both lower average errors and higher average classification accuracies, and also each meta-learner outperformed its counterpart using the Hyndman, et al. features with the exception of MetaDecisionTree with a higher 13.90% versus 13.75% sMAPE and a lower classification accuracy of 36.60 versus 43.33. However, all other meta-learners and meta-ensembles improve accuracy and corresponding ranks noticeably.

The decision tree algorithm plot allows an intuitive interpretation of both the splitting criteria used and the resulting node purity for the four classes of ETS base learners, as shown in Figure 2-3.

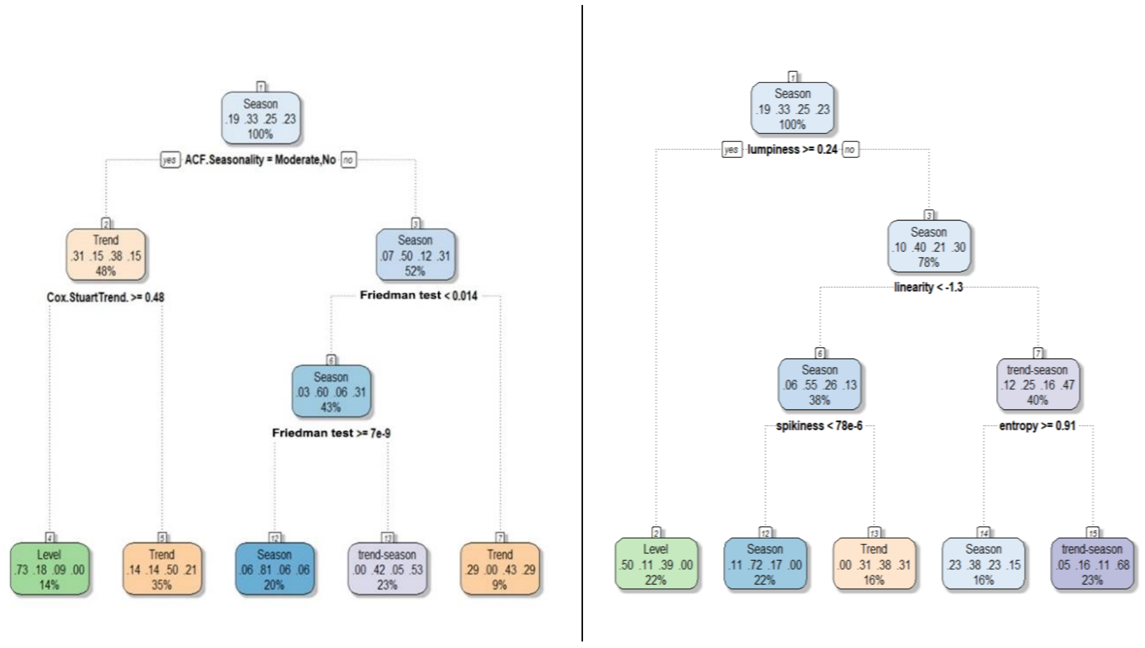


Figure 2-3. Plot of decision tree splitting criteria for Statistical Test and Hyndman, et al. meta-features (80-20% test)

The plot of the decision tree shows the first split is conducted on the ACF seasonality test, followed by p-value of Friedman test and p-value of Cox Stuart trend test. If the ACF Seasonality test says moderate or no seasonality (i.e. p>0.01), then Cox Stuart further splits with p-value >0.48 which rejects the trendy time series and label the time series as level, while p-value<0.48 has ended to trend. Alternatively, Friedman indicates no seasonality (with p >0.014) and the Friedman test further splits into strong (i.e. p<0.01) or not strong. Apparently, most splitting criteria focus on differentiating seasonality with multiple test which helps to explain why meta-learning outperforms simple statistical seasonality tests. Note that the tree-structures show only the top features used after the algorithm automatically prunes the tree to a moderate size and level of generalisation, in essence performing feature selection on the feature sets as part of the parameterisation. The metaDecisionTree built on the Hyndman-features on the right naturally utilises different features to split, most prominently lumpiness, followed by linearity, spikiness, and entropy, not including any features suitable to indicate seasonality, trend or constant patterns.

To derive additional insights, an export of the relative importance of the input features, measured by the mean decrease in Gini for the metaRandomForrest, provides the order of the most to least discriminant features.

For time series tests, the seasonal tests of Friedman (6.8916) is most important, followed by ChiSq seasonal test (5.0259), Kruskal Wallis (4.7067) and Friedman test 2 (4.5204), Cox Stuart dispersion test (3.9875), Pearson multiplicative pval (3.3965) and Spearmans Rho (3.1352), moderate importance of otherwise important test Cox-Stuart Trend (2,8755), Mann Kendall (2.6623), Linear Coefficient (2.5234) and p-value of ADF (2.5122), ACF Seasonality (1.8424), with most stationarity tests trailing the others, with KPSS (1.6176), ADF Constant-Trend (1.469), ACF Stationarity (1,3005065), ADF Constant-No Trend (1.2959) and ACF Trend (1.2785) as expected. For the metaRandomForrest split on meta-features from Hyndman et al., the analysis of the mean decrease in Gini indicates the different features used, but notably also starting with the feature of season (5,971061), but then followed by seemingly less relevant attributes such as lumpiness (5,674215), linearity (4,982886), curvature (4,344528), trough (4,226755), variance change (4,000030), KLscore (3,715280), change.idx (3,401562), Cpoints (3,380744), spikiness (3,345100), peak (3,286640), level shift (2,985979), entropy (2,967787), and the presumably relevant indicators of trend (2,646244), ACF1 (2,329272) and flat.spots (1,812584) trailing other features less intuitively related to the correct selection of a base learner.

Overall, in comparison to individual or aggregate model selection approaches, both sets of meta-features outperform all alternative approaches (see errors and accuracy in Table 2-4 of the previous section) indicating the suitability of meta-learning irrespective of the meta-feature set to differentiate patterns for the selection of base learners. With meta-features of statistical tests outperforming the Hyndman-features these should be considered for future use, or potentially combined with additional features for an even stronger feature set.

2.5.3 Meta-learning robustness by data conditions

Next, we seek to assess the robustness of the meta-learning approach conditional on the data conditions, in particular the split between training data provided to the meta-learner for parameterisation and the test data for out-of-sample evaluation of its accuracy. Given the limited number of 111 time series in the dataset, any allocation of more data to training may void the ability to generalise on fewer time series of the test data. We rerun all experiments on different splits of the hold-out data, gradually altering the distribution from 80% training and 20% test data split to 70%-30% and 60%-40% split. As this changes the ground-truth of the datasets, we also provide RelsMAPE errors and classification accuracy for all benchmark algorithms of individual selection, aggregate selection and statistical tests in Table 2.5-3. Classification and Forecasting Results of Different Meta-Learning Dataset split.

Table 2.5-3. Classification and Forecasting Results of Different Meta-Learning Dataset split

	40%-60% (test – train)			30%-70% (test – train)			20%-80% (test – train)		
	RelsMAPE		Accuracy	RelsMAPE		Accuracy	RelsMAPE		Accuracy
	Train	Test	Test	Train	Test	Test	Train	Test	Test
selMinError	0.865	0.841	-	0.853	0.850	-	0.824	0.873	-
metaDecsionTree	0.919	0.890	47.62	0.882	0.929	43.75	0.882	0.933	36.60
meta NN	0.919	0.907	42.86	0.882	0.929	53.12	0.882	0.933	46.67
metaSVM	0.939	0.901	45.24	0.882	0.929	43.75	0.882	0.933	53.33
metaRandomForrest	0.865	0.907	47.62	0.882	0.929	40.62	0.824	0.933	60.00
metaBagTree	0.865	0.951	47.62	0.882	0.929	46.88	0.824	0.933	63.33
metaXGBoost	0.865	0.907	35.50	0.882	0.929	47.22	0.824	0.933	56.66
selAIC	0.980	0.967	45.00	1.000	0.929	48.10	0.941	0.933	53.33
selETS(ZZZ)	0.973	0.940	47.62	1.000	0.929	44.12	0.941	1.000	44.00
aggETS(ANN)	1.162	1.099	19.00	1.118	1.286	14.00	1.059	1.200	14.00
aggETS(AAN)	1.270	1.099	12.00	1.118	1.429	10.00	1.176	1.200	12.00
aggETS(ANA)	0.973	0.951	34.00	0.941	1.000	38.00	0.941	1.000	38.00
aggETS(AAA)	1.000	1.000	35.00	1.000	1.000	40.00	1.000	1.000	36.00
Naive	1.257	1.110	35.00	1.176	1.429	37.00	1.353	1.467	35.00
Seasonal Naive	1.128	1.069	20.00	1.059	1.214	25.00	1.000	1.333	22.00

testCox-Stuart & Friedman	1.007	0.934	20.00	0.941	1.143	0.17	0.941	0.933	40.00
test Mann-Kendall & Friedman	1.007	0.934	15.55	0.941	1.143	25.00	0.941	1.000	30.00
testCox-Stuart & Spearman	1.101	0.956	28.89	1.000	1.214	25.00	1.000	1.067	40.00
testCox-Stuart & Kruskal-Wallis	0.986	0.918	28.88	0.941	1.071	30.56	0.941	0.933	46.67
Mean MetaTests	0.895	0.910	44.41	0.882	0.929	45.89	0.824	0.933	52.77
Mean Sel	0.976	0.953	46.31	1.000	0.929	46.11	0.941	1.000	44.00
Mean Agg	1.101	1.037	25.00	1.059	1.143	25.50	1.059	1.067	28.67
Mean Tests	1.026	0.936	23.33	0.941	1.143	20.18	0.941	1.000	39.17

	Rank Train	sMAPE Test	rkAcc Test	Rank Train	sMAPE Test	rkAcc Test	Rank Train	sMAPE Test	rkAcc Test
metaDecsionTree	4	1	1	5	1	6	5	4	12
meta NN	4	3	7	4	6	1	4	4	6
metaSVM	6	2	5	5	2	6	6	7	4
metaRandomForrest	1	3	1	1	2	8	1	3	2
metaBagTree	1	10	1	1	7	4	1	1	1
metaXGBoost	1	3	8	1	2	3	1	1	3
selAIC	9	13	6	13	8	2	10	7	4
selETS(ZZZ)	7	9	1	11	2	5	7	12	8
aggETS(ANN)	15	15	16	16	16	16	16	15	17
aggETS(AAN)	16	15	18	17	18	17	17	16	18
aggETS(ANA)	7	10	11	10	9	10	7	13	11
aggETS(AAA)	11	14	9	14	10	9	13	10	13
testCox-Stuart & Friedman	12	7	14	9	12	18	12	9	9
test Mann-Kendall & Friedman	12	7	17	8	13	13	10	11	15
testCox-Stuart & Spearman	14	12	12	11	15	13	15	14	9
testCox-Stuart & Kruskal-Wallis	10	6	13	7	11	12	9	6	6
Mean Rank MetaTests	2.8	3.7	3.83	2.83	3.33	4.67	3.00	3.33	4.67
Mean Rank Sel	8.0	11.0	3.50	12.00	5.00	3.50	8.50	9.50	6.00
Mean Rank Agg	12.3	13.5	13.50	14.25	13.25	13.00	13.25	13.50	14.75
Mean Rank Tests	12.0	8.0	14.00	8.75	12.75	14.00	11.50	10.00	9.75

Analysing the relative errors, accuracies and their ranks within each dataset split we observe that meta-learners using statistical tests reliably outperform all alternative model selection approaches and show a constant relative ranking ahead of individual selection, selection using a combination of tests, with aggregate model selection last. This indicates the robustness of the meta-learning approach to changes in the data set and its ability to work with smaller datasets as well as larger datasets to select relevant features and classify time series.

Within meta-learners, we see an improvement in ensemble methods metaBagTree, metaRandomForrest, and metaXGBoost, showing lowest errors and highest classification accuracy on larger training datasets of 80% and 70%, towards a single learner of metaDecisionTree for less training data. At the same time, relative individual model selection performance deteriorates, and the use of simple test combination improves, whilst aggregate model selection performance remains inferior although. Note that the errors and classification accuracies across (random) dataset splits are no longer comparable, as they are computed on different underlying time series and thus patterns, and thus yield little insight as to the underlying dynamics of error improvements.

To further interpret the consistency of used features and their critical values in splitting the tree, Figure 2-4 visualises the splitting criteria used by the metaDecisionTree algorithms trained on each of the three different dataset splits.

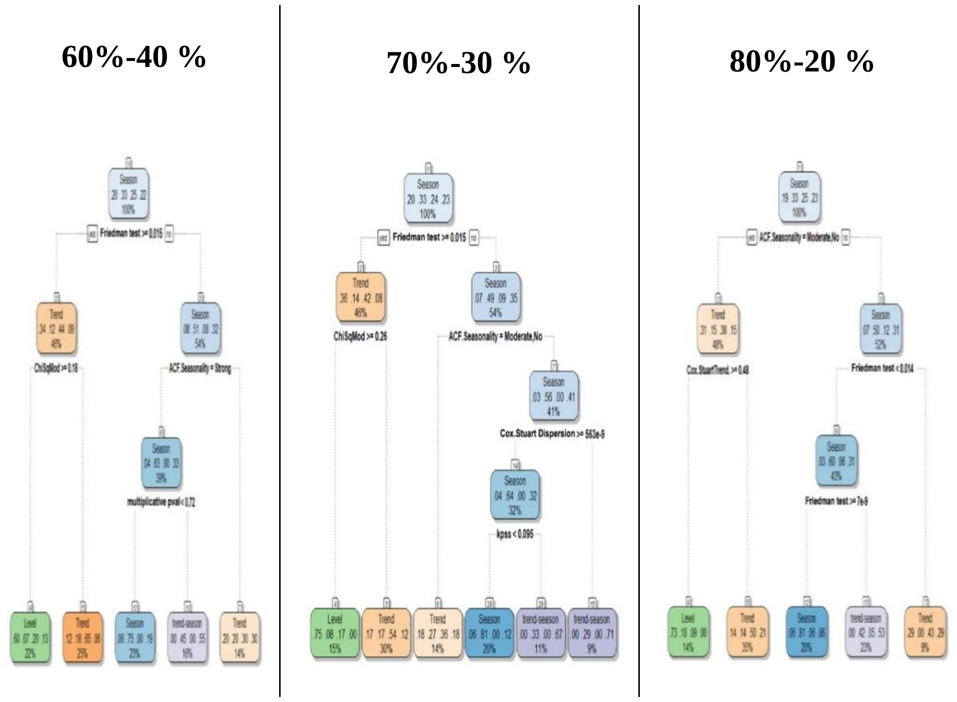

60%-40 %
Season .20 .33 .25 .22 100%
yes Friedman test >= 0.015 no
Trend .34 .12 .44 .09 46%
Season .08 .51 .08 .32 54%
ChiSqMod >= 0.18
ACF.Seasonality = Strong
Season .04 .63 .00 .33 39%
multiplicative pval < 0.72
Level .60 .07 .20 .13 22%
Trend .12 .18 .65 .06 25%
Season .06 .75 .00 .19 23%
trend-season .00 .45 .00 .55 16%
Trend .20 .20 .30 .30 14%
70%-30 %
Season .20 .33 .24 .23 100%
yes Friedman test >= 0.015 no
Trend .36 .14 .42 .08 46%
Season .07 .49 .09 .35 54%
ChiSqMod >= 0.26
ACF.Seasonality = Moderate,No
Season .03 .56 .00 .41 41%
Cox.Stuart Dispersion >= 563e-9
Season .04 .64 .00 .32 32%
kpss < 0.095
Level .75 .08 .17 .00 15%
Trend .17 .17 .54 .12 30%
Trend .18 .27 .36 .18 14%
Season .06 .81 .00 .12 20%
trend-season .00 .33 .00 .67 11%
trend-season .00 .29 .00 .71 9%
80%-20 %
Season .19 .33 .25 .23 100%
yes ACF.Seasonality = Moderate,No no
Trend .31 .15 .38 .15 48%
Season .07 .50 .12 .31 52%
Cox.StuartTrend. >= 0.48
Friedman test < 0.014
Season .03 .60 .06 .31 43%
Friedman test >= 7e-9
Level .73 .18 .09 .00 14%
Trend .14 .14 .50 .21 35%
Season .06 .81 .06 .06 20%
trend-season .00 .42 .05 .53 23%
Trend .29 .00 .43 .29 9%

Figure 2-4. Decision Tree plots for data splits of 60%-40%, 70%-30%, and 80%-20% (from left to right)

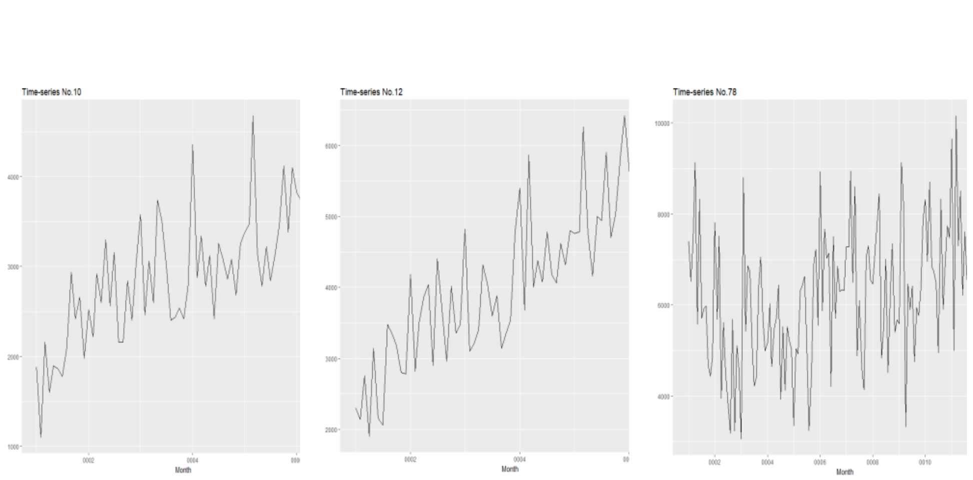
The metaDecisionTree trained on 60%-40% training and test data split, splits first on the Friedman test, then ChiSq Seasonality test and ACF seasonality test, and finally on Pearson multiplicative seasonality test which all are seasonality tests. metaDecisionTree trained on a 70%-30% data split, providing more evidence in training data to construct the splitting rules yet less data to evaluate it; the Friedman seasonality test is chosen first, followed by the ChiSq seasonality test and the ACF seasonality test, and employing the Cox-Stuart dispersion test followed by a KPSS stationarity test. Finally, as already mentioned, the 80%-20% split first splits on an ACF seasonality tests to separate classes “no”, weak” and “moderate” seasonality (i.e., p>0.01) from “strong” (p<0.01), followed by the Cox-Stuart Trend and Friedman seasonality test run twice using different thresholds next.

We observe that as the data split changes the meta-learner changes the selected features, their order (and thus relative importance) and the thresholds depending on the provided training data (sensitivity to data). However, some consistent patterns emerge: first, all meta-learners utilise seasonality tests, indicating their predictive power in reducing forecast errors; second, all meta-learners combine multiple seasonality tests, indicating that the use of a single seasonality test is not sufficient to reduce forecast errors. This also justifies the relative performance, showing how meta-learning is capable of outperforming a single combination of statistical seasonality tests. Indeed, most splitting criteria across metadata samples aim at classifying seasonality, which suggest that classifying seasonality correctly is of particular importance in improving accuracy on the NN3 dataset, as was indicated by the competition organisers. Amongst the tests, the ACF seasonality test and the Friedman seasonality test are included as meta-features in all meta-learners, pointing to their predictive power. In comparison, trend tests show less predictive value, with only the Cox-Stuart Trend utilised at lower levels of the decision tree.

2.5.4 Limitations in selection accuracy

To exemplify challenges in determining the ground truth, we seek to show limitations in matching the true class membership based on the ex-post actual forecast errors with the time series data based on selected examples. This should be interpreted as an attempt to show the challenges always inherent in model selection on empirical datasets with high noise levels, outliers, and structural breaks, and not to undermine the validity of applying classification on time series data.

Selected time series are plotted in Figure 2-5, with their true class membership versus the predicted class membership by each meta-learner provided in Figure 2-5.



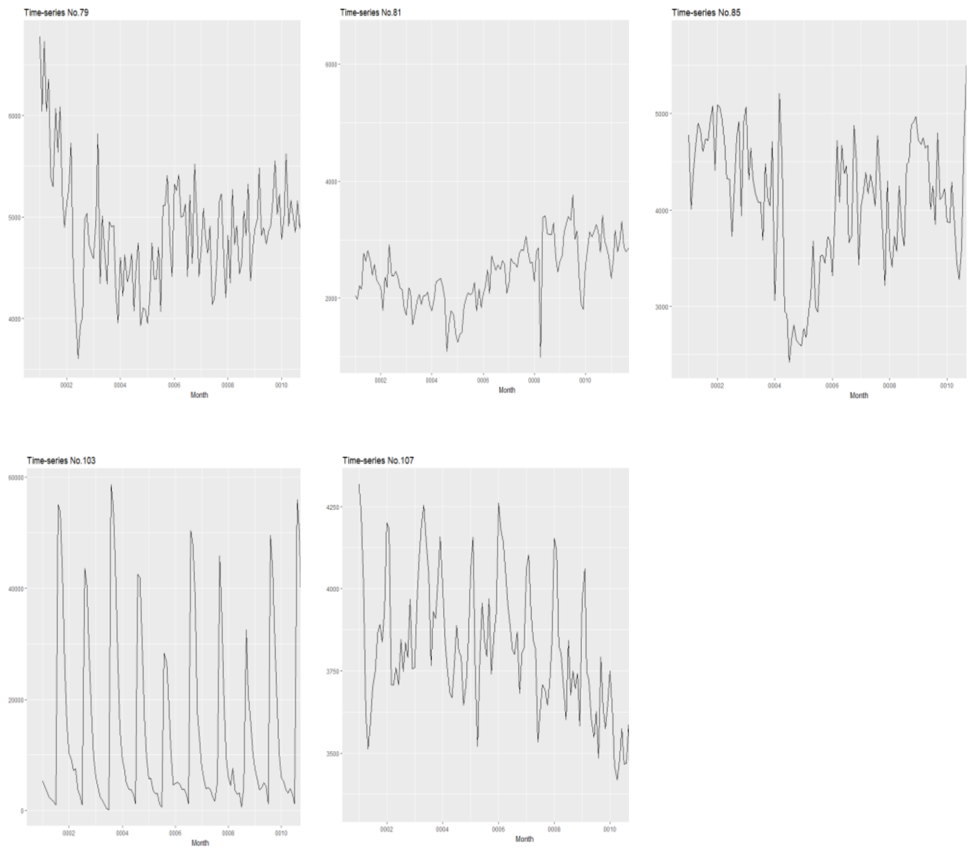


Figure 2-5. Plot of 8 time series with complex patterns and structural breaks

Many of these series show structural breaks, such as flattening out trends in time series #10, #78 and #107, or level shifts in #81, as well as outliers in series #79 and #85, all contained in the test data. Other patterns such as seasonality in series #78 is masked by a high level of randomness. As a result, the associate ground truth assigned to each series (identifying the "true" time series pattern by proxy of selecting the ETS algorithm and thus model form with the lowest test errors). Consider series #78: with randomness overshadowing seasonality, a constant ETS variant (ANN) shows a lower forecast error on the test data than a potentially more adequate seasonal variant, which has similar but slightly higher errors. It stands for discussion whether the true class membership should now be the one of the observed patterns, identified by a statistical expert using additional tools such as autocorrelation and partial autocorrelation functions (ACF/PACF), or by the base learner providing the lowest forecast accuracy using a robust fixed-horizon multi-origin evaluation with a robust error metric. These potential discrepancies on a subset of series are verified in Figure 2-5.

Table 2.5-4. The prediction of meta learning for randomly selected time series vs best forecast

	Xgboost	NN	DTree	RF	BagTree	SVM	Best Forecast
TS, No 10	Trend	Season	Season	Trend	Trend	Season	Trend
TS, No 12	Trend	Season	Trend-Season	Trend	Trend	Season	Trend
TS, No 78	Season	Season	Season	Season	Season	Season	Level
TS, No 79	Level	Trend	Level	Level	Level	Seasonal	Trend
TS, No 81	Level	Trend	Trend	Level	Level	Trend	Trend-Season
TS, No 85	Trend	Trend	Level	Trend	Trend	Seasonal	Season
TS, No 103	Season	Season	Season	Season	Season	Season	Trend-Season
TS, No 107	Season	Trend-Seasonal	Trend-Season	Trend-Season	Trend-Season	Season	Season

However, attempts to employ an alternative (manual expert based) labelling approach for the ground truth showed equal challenges, in improving classification accuracy, but also increasing forecast errors. As a result, we alert the reader to limitations in identifying ground truth on non-synthetic time series with structural breaks, dominant outliers and noise.

2.6 Conclusion

Partial or full automation in time series model selection with increasing pace in the number of time series and power of computing, as well as complexity in the dynamic pattern of time series is an inevitable requirement. It is a common knowledge that no model can obtain high accuracy on all data set, therefore, discovering the pattern in which the algorithm can perform well is really valuable. Feature-based time series representation (meta-features) aids to enhance our interpretation about the structure and properties of time series and ease the decision-making process for selection of time series prediction models. Meta-features provide a guideline of how to relate a learning algorithm with those domains in which an algorithm performs well. A large number of meta-features have been discussed in different studies; however, using statistical tests, which have a solid background in time series evaluation, to assess their efficiency as meta-features in meta-learning approach is relatively neglected, and there is limited research for investigation of their impact on creating meta-learning. Therefore, three main ideas have been studied in this chapter. Firstly, a review about the model selection approaches in time series forecasting is presented; secondly, the feature-based time series representation and its literature have been investigated; and finally, the effectiveness of using statistical tests as new meta-features is evaluated, and the relationship between the meta-features and the meta-learner algorithms have been clarified.

The point we should pay attention to is that our goal in this chapter is not finding the best forecasting algorithm. In other words, we are not going to implement a huge number of forecasting algorithms to find the best one. Rather, our goal is to check whether the meta-learning method as a filter approach can predict four patterns: level, season, trend, and trend-season. As mentioned, we used ETS algorithms as forecasters in the meta-learner. We do not use ARIMA algorithms in our research, but some of the exponential smoothing models are special cases of ARIMA models. Therefore, the study can be extended to the family of ARIMA methods without significant differences in the results.

In general, choosing the base learners and meta-features need intelligent design. The extracted meta-features in all meta-learning problems not only should be representative for their problem domain, but also to the base forecasting models. For example, in the current chapter, we use statistical tests which can measure the forecast models' properties (e.g., Trend, Seasonality). In the literature, none of the studies justifies their reasons for choosing the base forecast combination.

Future research directions of the chapter include but are not limited to:

1. Using ensemble meta-learners in the framework of fusion models to improve the classification accuracy,

2. Applying more forecast algorithms as base learners and evaluate their behaviour regarding the statistical test,

3. Increasing the number of meta-features and focus more on data-driven meta-features

4. Creating features based on expert’s knowledge for the problem at hand

5. Assessing the potential of the proposed model on different time series data sets such as M3, M4, and NNGC,

6. Considering feature selection methods for weighting the meta-features or selecting the most important ones.

3 On the Design of Meta-learning for Forecast Selection

3.1 Introduction

Time series forecasting is important for business analytics. Accurate forecasts can help enterprises make informed business decisions. How to select a suitable forecasting model or a suitable combination of forecasting models is at the core of forecasting. For real data, we do not know the underlying data generating process, so the modeller has to choose from the multitude of available models and methods, each appropriate to capture different characteristics in a time series. The problem itself is difficult and becomes more challenging when the potential models are numerous, the number of time series large, or the sample size of each series limited.

There are several approaches to perform model selection for forecasting problems. It can be done as a wrapper-based approach, where all candidate models have to be evaluated, and the selection is based on their performance by using information criteria (Burnham and Anderson, 2002) or empirical cross-validated error metrics (Fildes and Petropoulos, 2015) or judgment (Petropoulos et al., 2018). Regardless of its simplicity, this approach requires substantial computation cost, especially when we have multiple models and a large number of time series. On the other hand, we have filter-based approaches, which implement some rules to filter out models to reach the best model for each type of series without applying all the candidate models (Box et al., 2015). Filter-based model selection can also be performed effectively using expert judgment (Goodrich, 1992, Adya et al., 2001). Although by implementing the expert insights we may reach better results, the availability of experts and cost-related issues can limit the scope of this approach both for wrappers and filters. Alternatively, classification models can be used to learn the relation between a time series and model performance, so as to aid the selection. This approach, which is called meta-learning, incorporates a learner model (meta-learner) to find out the connection between time series characteristics and model performance, and overcoming some of the limitations of conventional model selection.

In meta-learning methodology, choosing distinct data characteristics (called meta-features) as inputs for the meta-learner is a critical and challenging stage, and its performance relies strongly on the appropriateness of the selected meta-features. Meta-features that abstract the structure of data can be created using descriptive statistics (like standard deviation, mean, skewness, granularity, etc.), model-based methods (like serial correlation, nonlinearity measures, largest Lyapunov exponent, etc.), statistical tests (like Cox-Stuart dispersion test, Spearman's rho test, Noether's cyclical trend test, Kruskal-Wallis test, F-test, etc.; Vilalta et al. 2004), among other approaches.

Most proposed meta-learning methods depend on the constructed meta-features selected by the modeller. However, there is limited research on evaluating the effect of using alternative banks of expert-designed features on the performance of the meta-learning. In this paper, we put more attention on the effectiveness of different sets of meta-features. More precisely, we group meta-features gathered from different sources based on their method of calculation and compare their relative contribution in forecasting model selection accuracy to uncover the effect of each group. Moreover, we propose a novel data-driven meta-features extraction approach using deep unsupervised learning for automatically generating the rich meta-features and compare the most effective manual selected features with our proposed data-driven meta-features.

In order to extract the interaction between meta-feature, forecasters, and meta-learners, different tree-based classifiers are implemented as potential meta-learners. We use decision tree, random forest, and XGBoost classifiers to investigate the sensitivity of our findings. We also explore the impact of the pool of forecasts that the meta-learner chooses from. We include simple and complex ones, the latter containing automated model building, or selection algorithm and combination methods. Finally, in a large dataset, we compare the performance of model combination and model selection separately and show even a simple meta-learning outperform the individual model selection.

In order to provide a comprehensive package to facilitate the process of meta-learning in forecasting time series and to conduct various investigations, we introduce the MetaTS package, a free and open-source Python library developed to ease meta-learning for time series forecasting by offering a toolkit containing the typical components needed for a meta-learning workflow. In addition to providing new components and facilities, I aim to unify the available Python libraries which can be useful for meta-learning on time series data.

The contributions of this study can be summarised as follow:

- Comparing different groups of meta-features in the meta-learning context, and exploring if particular classes of meta-features are more informative
- Proposing a new group of data-driven meta-features using deep unsupervised learning and compare them with manually selected features
- Evaluating the effect of using simple and complex base forecasts on the performance of the meta-learner and the usefulness of meta-features
- Analysing the effect of meta-learners on the overall performance of meta-learning and comparing the model selection versus model combination
- Proposing a free and open-source Python library to ease meta-learning for time series forecasting
- Analysing the meta-features importance in the meta-learning

The rest of this chapter is organised as follows. Section 3.2 summarises the relevant model selection literature. In Section 3.3, we detail the meta-learning used here. Section 3.4 outlines the experimental design and Section 3.5

presents our results, which are discussed further in Section 3.6. In Section 3.7, we propose a new data-driven meta-learning and discuss the efficiency of automated feature extraction and finally, conclusions and further research directions are provided in Section 3.8.

3.2 Literature review

There are multiple approaches to addressing model selection in the forecasting literature. Typically, this has been done by comparing competing forecasts using information criteria or empirical cross-validated errors.

Using information criteria, we can devise automated procedures for model selection (Hyndman et al., 2002). Several alternative information criteria can be used, for example, the Akaike’s Information Criterion (AIC) or the Bayesian Information Criterion (BIC). These penalise the model fit that has the maximised likelihood, for its complexity, as captured by the number of parameters (Burnham and Anderson, 2002). Selecting models using different information criteria is shown to result in potentially different models but with similar results from a forecasting performance perspective (Billah et al., 2006). Nevertheless, when using information criteria, the likelihood function should be comparable across all models. Accordingly, comparing models that use different data transformations can be challenging, and precludes choosing among non-model-based forecasters, which causes problems when diverse comparisons of forecasts are needed.

Using a validation set to compare the performance of forecasts is another common approach for model selection (Fildes and Petropoulos, 2015). After dividing data into training and validation sets, and then fitting and evaluating forecasts on them, the best performing method on the validation set is chosen. This procedure relies on measuring the forecasts’ performance multiple times, using a rolling origin evaluation scheme. The idea behind the rolling origin scheme is to train and validate a model on different subsets of data, much like with cross-validation. However, in forecasting, for many methods and models, the temporal ordering is important, and therefore with rolling origin we always move forward in time, in contrast to the standard cross-validation, as we roll across forecast origins in the validation set. Evaluating models with rolling-origin can increase the robustness of predictions and reduce the sensitivity to a specific subset of data (Tashman, 2000, Fildes, 1992). Although AIC is equivalent to the one-step-ahead out-of-sample cross-validated mean squared error (Stone, 1977), cross-validation can be applied with different forecast horizons and error metrics to obtain results corresponding more to the problem at hand. However, the cross-validation approach, similar to employing information criteria, introduces substantial computational cost for selecting the best model, as not only we need to generate all forecasts, but also iterate them across the rolling origins.

An effective alternative to information criteria or using a validation set is meta-learning. The meta-learning concept was presented in the context of forecasting time series by Prudêncio and Ludermir (2004). Meta-learning consists of different elements: the meta-features, the meta-learner, and the base forecasters. The meta-learner is tasked with capturing the relation between extracted features of time series (meta-features) and the forecasting performance of the base forecasters. Accordingly, the learned knowledge can be used to predict the best forecaster for each time series based solely on its characteristics.

Although the classifier (meta-learner) has an essential role in meta-learning, capturing a high-quality representation of the time series behaviour by identifying the best meta-features set has been attracted the most attention in meta-learning studies. Three main classes of meta-features have been suggested: (i) statistical and information-theoretic characterisation, (ii) model-based features, and (iii) landmarkers (Brazdil et al., 2008). The first group estimates the statistical features of the dataset, for example, the mean, standard deviation, skewness, kurtosis, and length of the series. As an example of the second type, one can build a decision tree from a dataset and capture the properties of the tree, such as the maximum tree depth, its shape, and the tree imbalance. Finally, the last class of so-called landmarkers exploits information obtained from the performance of a set of forecasts to characterise the time series. The differences between the model-based features and landmarkers are related to the fact that the latter does not come under model characteristics, but rather performance measures. Kück et al.

(2016b) used errors on the training, validation, and test sets of the neural network as landmarkers. In the literature, there have been several studies that explored the different combinations of various time series features in meta-learning as their model selection approach.

Prudêncio and Ludermir (2004) empirically explore the feasibility of meta-learning in time series forecasting model selection using two case studies. In the first study, they introduce two forecasting models (a time-delay neural network and simple exponential smoothing model) and choose ten meta-features including length, autocorrelation coefficients, statistically significant coefficients, coefficients of variation, skewness, kurtosis, and a test of turning points for randomness, from 99 time series, and then employed the C4.5 Decision Tree algorithm as a meta-learner. They find that the average error of forecasts proposed by the meta-learner for the 99 test time series is lower than that of using either of the two forecasters. In the second study, they chose another batch of candidate forecasting models (i.e. random walk, Holt's linear exponential smoothing, and AR models) and increased the number of time series to 645 yearly time series from the M3 competition (Makridakis and Hibon, 2000). Besides selecting forecasting model, their proposed meta-learner NOEMON method can output the ranking of the forecasting models (Kalousis and Theoharis, 1999).

Similarly, Wang et al. (2009) proposed a meta-learning framework to recommend suitable models to generate forecasts. They introduce a set of more comprehensive time series features, and the number of candidate forecasters is expanded to four, including three traditional statistical models and one machine learning algorithm (a Neural Network). They use both supervised (C4.5 Decision Tree) and unsupervised (SOM clustering algorithms) meta-learners and visualise the time series features in a two-dimensional map. Their results indicate that using combined forecasts, weighted based on their derived selection rules, can achieve better forecasting accuracy compared to using only one of the base forecasters. Inspired by Wang et al. (2009), Lemke and Gabrys (2010a) further explore the performance of the rank-based combination of candidate forecasting models and compare it with different meta-learners, such as a Neural Network, a

Decision Tree, and a Support Vector Machine. They conclude that this combination method can achieve a comparable result over the selection method. Due to the generality and usefulness of the time series features introduced by Wang et al. (2009), the set of features are adopted by Widodo and Budi (2013). In an attempt to further reduce the dimensionality of the extracted time series features, they use principal component analysis. They find that such a dimensional reduction process can be helpful for improving the forecasting performance of meta-learning without damaging its classification performance.

Kück et al. (2016b) consider a new set of meta-features related to forecasting errors for candidate forecasters and employ a Multilayer Perceptron (MLP) as the meta-learner. They use their meta-learning framework in a large number of time series related to industry sales from the NN3 competition (Crone et al., 2011) and find that including error-based features can help the meta-learner capture more useful meta-knowledge related to the forecasting models' performance and thus help improve the overall accuracy of forecasting. Cerqueira et al. (2017) introduce a meta-learning framework where a so-called arbitrated dynamic ensemble, the meta-learner, is used to predict the weights for candidate forecasters based on their prediction errors. These are combined to generate forecasts based on their weights. Their results show that this framework obtains similar performance compared to other state-of-the-art methods for combining forecasts.

Talagala et al. (2018b) propose a more general meta-learning framework called FFORMS (Feature-based FORecast Model Selection) where a Random Forest algorithm is adopted as meta-learner to separately learn the selection mapping using meta-features extracted from yearly, quarterly, and monthly time series from the M1 (Makridakis et al., 1982) and the M3 competition (Makridakis and Hibon, 2000). They further expand the feature space to include 33 time series features based on Wang et al. (2009); Hyndman et al. (2015a); Kang et al. (2017), from which 25 features are extracted from yearly time series while 30 features are extracted from quarterly and monthly time series. They also apply time series augmentation in the original time series set by simulating new time series which have similar characteristics to the sampled one. They show FFORMS to provide competitive results compared to a series of benchmark forecasting models.

Based on the FFORMS framework proposed by Talagala et al. (2018b), Montero-Manso et al. (2020) introduce a new meta-learning framework named FFORMA (Feature-based FORecast Model Averaging) where the output of the meta-learner is a set of weights assigned to each candidate forecasting model, instead of selecting only the best forecaster from candidate forecasting models. They further extend the time series space to include 100,000 time series, including daily, hourly, monthly, quarterly, and yearly time series from the M4 forecasting competition (Makridakis et al., 2020). They achieved the second most accurate point forecast and prediction interval performance in the M4 competition. Moreover, using a convolutional deep neural network, Ma and Fildes (2020) proposed a meta-learning framework to extract representative features from raw sales time series and implemented the obtained classification output to combine multiple base-forecasting models. They show that the proposed meta-learner provides superior forecasting performance compared with a number of state-of-art benchmarks.

By reviewing the literature in this field, we are curious to know the impact of different group of meta-features on the forecasting model selection and analysing the meta-features importance in the meta-learning.

3.3 Methodology

In this research, we aim to investigate the characteristics of meta-learning. To achieve this aim, we analyse the meta-learning from three facets: (i) the meta-feature; (ii) the meta-learner; and (iii) the base forecasters. For the first perspective, we use different groups of meta-features extracted from two main sources Tsfresh (Christ et al., 2018), and Tsfeatures (Hyndman et al., 2019). Moreover, we compare these groups with a proposed automated meta-features extraction approach using Autoencoders. Second, for the meta-learner, we used three classification algorithms of increasing capabilities. We also compare the effectiveness of model selection and combination in a big database. Finally, for the base forecaster, two different groups are used which can be categorised into simple and complex. We distinguish the complex forecasters as those that use some internal model selection or combination process. The aim of using these different viewpoints in studying these elements of meta-learning is to find how they affect the meta-learning performance, separately or together. Meta-learning Framework

The proposed meta-learning approach is shown in Figure 3-1. In the first step, single and complex forecasters are used to predict the target time series and ranked for each time series according to their accuracy. This is done on a training set of time series. Meta-features are extracted and computed for each of these time series. Afterwards, extracted features and forecasts' ranks are inputted into the meta-learner. The meta-learner identifies which base forecaster model can be considered as appropriate for a time series, depending on its structure and behaviour, as abstracted by the meta-features.

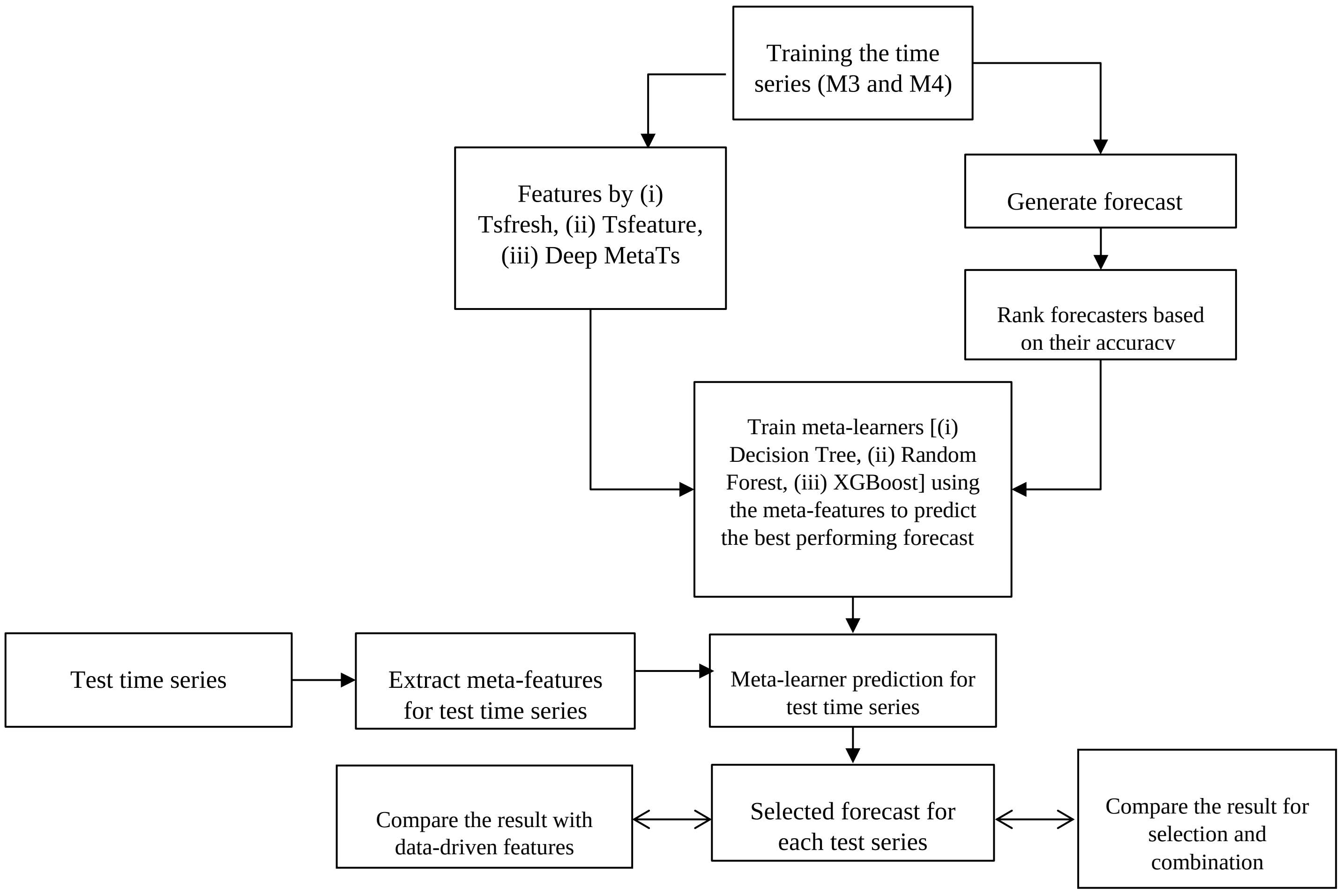

Training the time series (M3 and M4)
Features by (i) Tsfresh, (ii) Tsfeature, (iii) Deep MetaTs
Generate forecast
Rank forecasters based on their accuracv
Train meta-learners [(i) Decision Tree, (ii) Random Forest, (iii) XGBoost] using the meta-features to predict the best performing forecast
Test time series
Extract meta-features for test time series
Meta-learner prediction for test time series
Compare the result with data-driven features
Selected forecast for each test series
Compare the result for selection and combination

Figure 3-1. The proposed meta-learning methodology

For the time series in the test set, the meta-learners will predict which forecast is appropriate and will recommend it or combine the forecasters based on the meta-classifier weights. The process is repeated for all combinations of elements, resulting in 14 setups (two groups of meta-features, three classifiers, two sets of forecasts, a data driven group of meta-features, and a large data set for forecast selection and combination).

3.3.1 Meta-learners

- Decision Tree:

Decision trees are one of the more well-known classifiers, broadly studied in the literature. Decision trees are competitive to the other machine learning approaches not only in terms of prediction accuracy but also in model interpretability and simplicity (Brockwell and Davis, 2016). As classifiers, we consider the standard single decision tree, and the more advanced Random Forest, and XGBoost classifiers that are representative of single, bagged, and boosted versions of decision tree models, respectively.

- Random Forest:

Random Forest is an ensemble learning method that can be applied for both classification and regression (Liaw and Wiener, 2002). In Random Forests, multiple simple trees are generated, acting as weak learners that make use of a single or few features with limited tree size. The individual trees are then combined to produce the aggregate prediction of the Random Forest. In each splitting step at a tree’s nodes, a subset of features from the total features is selected. The tree split that reduces the maximum heterogeneity among sub-nodes is selected. Different measures have been proposed as metrics for evaluating the heterogeneity of nodes, including the Gini index, entropy, and the classification error (Hastie et al., 2009). The random selection of predictors enhances the classifier to be more unbiased and help it to avoid over-fitting. Furthermore, the use of limited features in each tree helps the classifier to train on high dimensional datasets effectively. Talagala et al. (2018b) used a Random Forest successfully in their meta-learner for the M4 forecasting competition (Makridakis et al., 2020).

- XGBoost:

XGBoost (Chen and Guestrin, 2016) is an ensemble method that incorporates a boosting method to integrate multiple weak learners and create a strong learner. It has been shown to perform very competitively in different forecasting and data mining competitions. In contrast to bootstrap aggregator (bagging) methods like Random Forest that ensemble weak models in a parallel approach, boosting models try to enhance true base models in a sequential process. At each iteration of the gradient boosting method, a new base model is trained to correct the previous predictor with the aim of optimising an appropriate loss function. A Regularisation term can be added to the loss function to increase the reliability of XGBoost (Zhang et al., 2018). Montero-Manso et al. (2020) applied XGBoost for meta-learning model combination in the M4 competition and ranked second in terms of forecasting accuracy.

We also implemented some other meta-learners such as simple neural network and SVM, but the results did not show an improvement in the forecasting performance and then we decided to only report the current models.

3.3.2 Base forecasters

As forecasting methods have increased in complexity, they often include some elements of model selection and combination. In this study, to investigate the effect of simplicity (or complexity) of the base learners, two different groups are tested. The first group of forecasters consists of a set of exponential smoothing models that are typically used in practice to cover different time series patterns. Exponential smoothing (ETS) models are classified based on their structure that can capture, with the inclusion of trend and seasonality components. These components can interact additively (A), or multiplicatively (M), between them and with the error term of the model. The trend can be linear or damped (d). Using the model classification of the space-state ETS models by Hyndman et al. (2008), different models are designated with three parts. The first part stands for the type of the error term ("A" or "M"), the second part is for the trend ("N", "A", "Ad", "M" or "Md"), and the last one is for the type of seasonality ("N", "A", or "M"). In all cases "N" stands for none, characterising the case when there is no trend or seasonality. We choose ANN, AAN, AAdN, MNM, and MAM from the ETS family of models as simple base forecasters. These correspond to some of the well-known exponential smoothing methods, such as the single exponential smoothing (ANN), Holt’s smoothing (AAN), Damped trend (AAdN), and two refined versions of Holt-Winters smoothing without and with a trend (MNM and MAM) that assume multiplicative errors. These groups of models are common in practice and approximate many of the other specifications in the ETS (Weller and Crone, 2012). We also use the Theta method (Assimakopoulos and Nikolopoulos, 2000), which was the best performer in the M3 forecasting competition (Makridakis and Hibon, 2000), and can be seen as a special case of exponential smoothing (Hyndman and Billah, 2003).

The second group of forecasters contains complex forecasters. Complexity means that the learning method is complex, or the setup of it (such as with many hyperparameters or other tuning/design choices). Automated ETS (Hyndman et al., 2008), Automatically specified ARIMA, MAPA (Kourentzes and Petropoulos, 2014), TBATS (De Livera et al., 2011), forecTheta (Fiorucci et al., 2016), and Temporal Hierarchical Forecasting (THieF) (Athanasopoulos et al., 2017) are considered as complex base forecasters. These are detailed below.

- Automated Exponential Smoothing:

Automated exponential smoothing selects the best types of error, trend, and seasonality components automatically, by information criteria measures (such as AIC, or BIC). We follow Hyndman and Khandakar (2007) for automatically specifying the ARIMA models (Auto ARIMA). The order of differencing is determined first, relying on the statistical test, both for the level and seasonal differences. Subsequently, the orders of the autoregressive and moving average components are set by minimising the AIC corrected for sample size.

- Multi Aggregation Prediction Algorithm:

The Multi Aggregation Prediction Algorithm (MAPA) is a temporal aggregation methodology for the ETS models. In the first step, a time series with a given frequency is aggregated by considering groups of values to create constructed lower frequency representations of the time series. In the next step, each aggregated time series is predicted by an exponential smoothing model, which can differe based on the time series behaviour. In the final step, the obtained components of ETS models are combined to create a final forecast.

- TBATS:

TBATS belongs to the exponential smoothing family and includes an appropriate Box-Cox transformation, dampening, ARMA errors, and trigonometric seasonality (De Livera et al., 2011). TBATS can adjust the type of included components, similar to ETS, using information criteria.

- Force Theta:

The Theta method proposed by Assimakopoulos and Nikolopoulos (2000) uses a decomposition method that takes place on seasonally adjusted data. The variations and movements of time series are captured by using the fitted lines referred to as “theta lines”. ForecTheta is a generalisation of the Theta method that optimises the construction of theta lines, based on various validation schemes.

- Temporal hierarchical Forecasting:

The Temporal hierarchical Forecasting (THieF) methodology is proposed by Athanasopoulos et al. (2017) by generalising MAPA with the use of the hierarchical forecasting framework. THieF is more flexible than MAPA in that it is model independent and permits for most flexible schemes to combine the forecast across different aggregation levels. As with MAPA, THieF mitigates modelling uncertainty by relying on model combinations.

3.3.3 Meta-features

One advantage of using meta-features is their applicability to almost all-time series, disregarding their context, potential sub-structures, and other characteristics. Moreover, abstracting the time series modelling problem into a feature space may provide more insights about the behaviour of the time series, and improve the understanding of the relation between time series that use the same base forecaster (Harvey and Todd, 2015). Additionally, we note that the most crucial difference between highly ranked meta-learning methods in well-known competitions is linked to the creative selection of features (Blum and Hardt, 2015).

We utilise three built-in meta-feature packages designed for time series. The first feature set originates from a tool for time series feature extraction called Time Series Feature Extraction based on Scalable Hypothesis tests (Tsfresh). This facilitates the feature selection process for time series forecasting problems by combining 63 time series characterisation methods and calculates up to 163 features (Christ et al., 2018).

The second set, Tsfeatures, is provided by Hyndman et al. (2019) and includes 42 different methods for feature extraction. These features are gathered from different studies: Hyndman et al. (2015a), Kang et al. (2017), Talagala et al. (2018b), and Montero-Manso et al. (2020). All together this package consists of descriptive statistics and model-based meta-features including spectral entropy, autocorrelations, partial autocorrelation, nonlinearity, the strength of seasonality and trend, and so on.

Both mentioned packages proposed manual features that are not necessarily related to the time series generation process. On the other hand, for the first time, we used Autoencoders as a typical deep unsupervised neural network model to extract the rich features for meta-learning.

Autoencoders are unsupervised neural networks aiming to reconstruct input data (Hinton and Salakhutdinov, 2006). Autoencoders compress the input into lower-dimensional data in the encoder part and reconstruct the output from the encoded representation in the decoder part. The encoded data called the latent space is a compact representation or compression of the input.

More formally, in the context of autoencoders, there exist two spaces, which are often Euclidean spaces, decoded space X and the encoded space Z, while $X \in \mathbb{R}$, $Z \in \mathbb{R}$. Two main functions in autoencoders are encoder function $E_{\emptyset} : X \rightarrow Z$, which is parametrized by $\emptyset$ and the decoder $D_{\theta} : Z \rightarrow X$

parametrized by θ. The loss function, i.e., Mean squared error (MSE) for proposed autoencoders are defined as

$$\min_{\theta,\emptyset} L(\theta,\emptyset), where\ L(\theta,\emptyset) = \frac{1}{N}\sum_{i=1}^{N}\|x_i - D_\theta(E_\emptyset\ (x_i)\|_2^2 \qquad (3\text{-}1)$$

where D and E are the decoder and encoder, respectively.

The general architecture of autoencoders consisting of an encoder, latent representation and a decoder is shown in Figure 3-2.

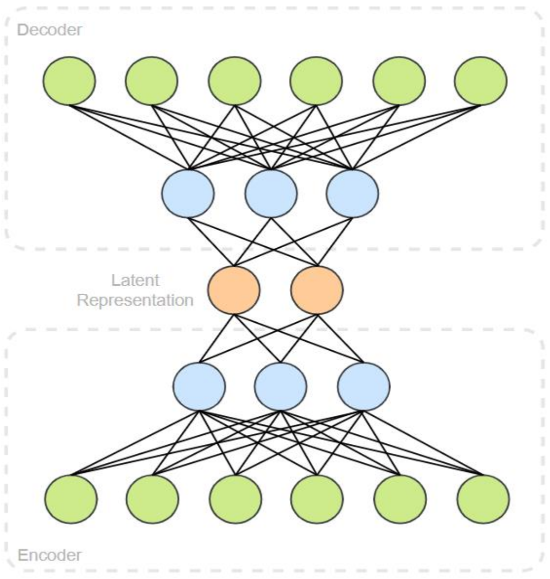


Figure 3-2. AutoEncoder Architecture

For the third group, we develop MetaTS which is an open-source Python library designed to ease meta-learning for time series forecasting by offering a toolkit containing the typical components needed for a meta-learning workflow. In addition to provide new components and facilities, MetaTS aims to unify the available Python libraries which can be useful for meta-learning on time series data. Here we use the meta-feature extraction part of this package to generate data driven meta-Features using deep autoencoders. For more information about the package, visit https://github.com/DrSasanBarak/metats

In this paper, we first focus on the mentioned predefined features from first and second packages, but we also use the autoencoder as a deep unsupervised learning model from third package to extract features from time series and compare them with the predefiend features. Of course, there are still features not considered like harmonics, fractalism, distributions of events, intermittent events, bursts of events, non-monotonicity/jumps, second order properties, derivate characteristics and others; however, MetaTs is supporting user-defined meta-features for implementing in future studies.

3.4 Experimental Design

3.4.1 Dataset

To perform our evaluation, we use the M3 and M4 competitions dataset (Makridakis and Hibon, 2000, Makridakis et al., 2018). The M3 competition data contains a total of 3003 time series of varying industries and sampling frequencies. Table 3.4-1 presents the forecast horizons and the number of time series for our training and test datasets.

Table 3.4-1. The classification of M3 data

Time Interval	**Horizon**	**Training**	**Test**	**Total**
Yearly	6	516	129	645
Quarterly	8	604	152	756
Monthly	18	1142	286	1428
Other	8	139	35	174
Total	308	2401	602	3003

We use a different forecast horizon for each set of time series, matching the design of the M3 competition. The horizon for the Yearly, Quarterly, Monthly, and Other series are 6, 8, 18, and 8 periods, respectively. With respect to the base forecasters, we evaluate the accuracy on a test set of equal length to the forecast horizon, using all other observations to fit the various base forecasters to the data. For each group of time series (yearly, quarterly, monthly, and other) we use cross-validation of 80% of the time series (2402 time series) to train the meta-learner and the remaining 20% (611 time series) to test the performance.

For a experiement on a larger data, we randomly select 10000 time series from the M4 monthly dataset to assess the performance of Meta-Learning. To obtain time series with equal length, we chop the beginning of each series. Furthermore, we rescale each time series as a pre-processing step, and eventually, we end up with 10000 series of length 120. Figure 3-3 shows some samples from the data. Different kinds of trends, seasonality, and other characteristics of time series is evident.

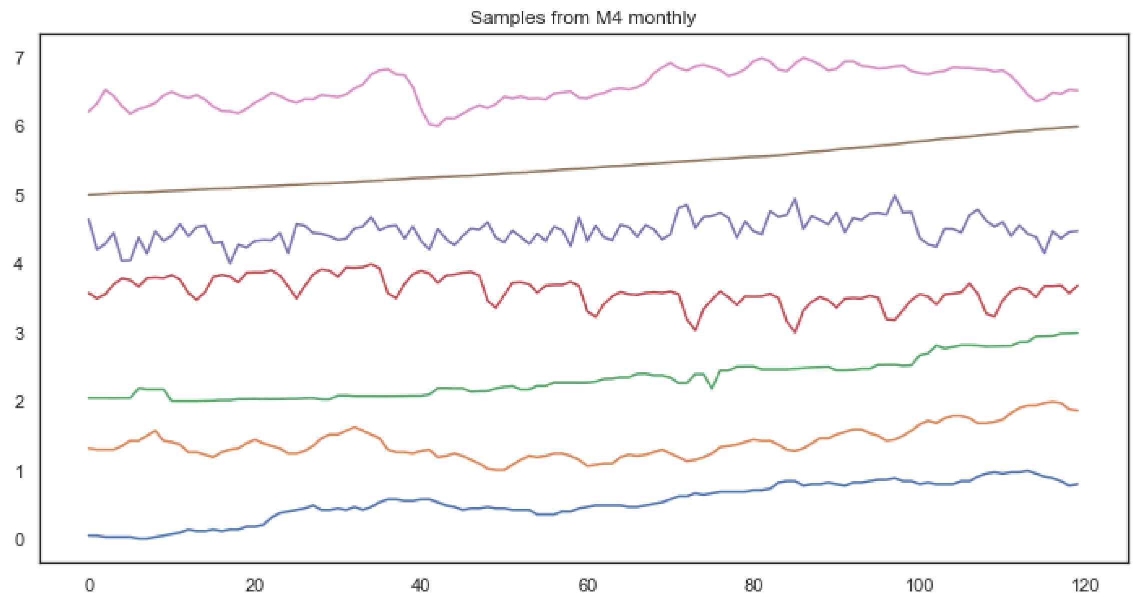


Figure 3-3. Sample of M4 monthly data

In our bigdata experiment, we divide each M4 series into three parts. The first part with a length of 108 used for training the meta-learner to forecast the second part i.e. 12 steps ahead. The last 12 data points is used as the test data to assess the performance of the meta-learner.

3.4.2 Base forecasters

3.4.2.1 Simple base forecasters

We used the 'smooth' package (Svetunkov, 2017) for the R language (Team, 2013) to produce four simple ETS models. The smoothing parameters, as well as any initial values, are optimised by maximum likelihood. Moreover, the forecasts for the Theta model are produced using the 'thetaf ()' function from the *forecast* package in the R language with its default configuration (Hyndman et al., 2020).

3.4.2.2 Complex base forecasters

In contrast to simple base forecasters, complex base forecasters have an internal model selection which helps to select better model form or hyperparameters according to time series structure.

For the automated exponential smoothing, that selects the best types of error, trend, and seasonality components automatically, we use the ‘es ()’ function from the ‘smooth’ package for R (Svetunkov, 2017).

For the ARIMA models, we use the ‘auto.ssarima()’ function from the same package.

For MAPA and forecTheta models, we use the R packages with the same name, MAPA and forecTheta, respectively (Kourentzes and Petropoulos, 2014, Fiorucci et al., 2016). For forecasting with TBATS, we use the ‘tbats ()’ function form ‘forecast’ package for R with its default configuration. Finally, we use the ‘thief’ package for R, using ARIMA forecasts for each temporal level, to produce the THieF forecasts.

3.4.3 Meta-learners

For implementing decision tree, random forest, and XGboost models, we use ‘rpart()’, ‘rf()’ and ‘xgbTree()’ functions from the Caret package for R (Kuhn, 2008), respectively. We use ‘repeatedcv ()’ from Caret package for controlling the training evaluating using 10-fold cross-validation repeated for five times with search approach as "random".

3.4.4 Meta-features

We use the Tsfresh python implementation (Christ et al., 2018), the Tsfeatures package (Hyndman et al., 2019) for R (Team, 2013), and our developed Python package, MetaTs. The extracted features, from both Tsfresh and Tsfeatures, are grouped with distinct categories of features based on their base calculation methods into autoregressive parameters, autocorrelation coefficients, wavelet transformation coefficients, Dickey-Fuller test parameters, energy-based properties, entropy-based features, Fourier Transform coefficients, time series quantiles, statistical summary values, STL decomposed features. However, the MetaTs provides a group of rich meta-features that is obtained from the process of time series generation.

The list of the individually extracted features with their corresponding group are reported in Appendix A. A summarised overview of each group of features is provided below.

3.4.4.1 Autoregressive parameters

These features are calculated based on an autoregressive (AR) model of past values of the variable. Let p be the maximum lag of the autoregressive process and y_t a time series values at time t, c_i coefficients in the following formula are considered as autoregressive features (Box et al., 2015).

$$y_t = c_o + \sum_{i=1}^{p} c_i y_{t-i} + \varepsilon_t \tag{3-2}$$

3.4.4.2 Autocorrelation coefficients

The autocorrelation measures the linear relationship between lagged values of a time series. There are several autocorrelation coefficients, based on the value of the lag. The lag k autocorrelation is defined as:

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \tag{3-3}$$

where $\bar{y}$ is the average values of y_t and T is the length of the time series (Box et al., 2015).

3.4.4.3 Wavelet transformation coefficients

Wavelet transformation modifies a signal (time series) into different forms with the aim of extracting hidden characteristics. In this study, we use values from different coefficients in a continuous wavelet transformation known as the “Mexican hat wavelet”. The transformation is defined as (Gomes and Velho, 2015):

$$y_{cwt} = \frac{2}{\sqrt{3a}\pi^{\frac{1}{4}}}\left(1 - \frac{{y_t}^2}{a^2}\right)e^{-\frac{{y_t}^2}{2a^2}} \tag{3-4}$$

where y_{cwt} is the continuous wavelet transformed time series and a is the width parameter of the wavelet function.

3.4.4.4 Dickey-Fuller test parameters

The Dickey-Fuller test is a hypothesis test that checks whether a unit root is present in a time series sample. This group of features returns the values of the respective test statistic.

3.4.4.5 Energy-based properties

The sum over the squared values of a signal (time series), which is called" energy", is representative of energy carried in the signal. Absolute energy and energy ratio by chunks (energy of a subset of the time series) are considered as the energy-based properties for time series (Pollock et al., 1999).

$$E = \sum_{t=1}^{T} {Y_t}^2 \tag{3-5}$$

3.4.4.6 Entropy-based features

In information theory, the inherent uncertainty in the possible outcomes of a random variable is referred to as entropy. Different values can be calculated based on the entropy concept, including Shanon entropy (Shannon, 1948), approximate entropy (Yentes et al., 2013), and sample entropy (Richman and Moorman, 2000). These are implemented as entropy-based features.

3.4.4.7 Fourier Transform coefficients

Fourier coefficients of a one-dimensional discrete Fourier transform for real inputs are obtained by the Fast Fourier Transformation algorithm to create these groups of features. A Fast Fourier Transformation is defined as (Phillips et al., 2003):

$$C_K = \frac{1}{N\sum_{t=0}^{N-1} y_t \exp\left(-2\pi i \frac{tk}{N}\right)} \qquad for \quad K = 0 \text{ to } N-1 \tag{3-6}$$

where *N* is the number of all periods, y_t is the value of the series at time *t*, *k* is the current frequency we are considering, and C_k is the value of the meta-feature at frequency *k* in the time series. These are used as features.

3.4.4.8 Time series quantiles

This group of features contains values represents the conditional distribution of time series, including sample q quantile (value of time series greater than q% of time series ordered values), the change quantile (difference among two quantiles), and other variables.

3.4.4.9 Descriptive statistics

Descriptive statistics of the time series such as its mean, variance, minimum, and maximum.

3.4.4.10 STL decomposed features

The STL method was developed by Cleveland et al. (1990) as a robust method for decomposing time series into seasonality, trend, and an error component. Various measures of trend and seasonality of a time series based on the STL decomposition are provided.

3.4.5 Evaluation metrics

The accuracy of the different forecasting approaches is evaluated using the Average Relative Mean Absolute Error (AvgRelMAE). The AvgRelMAE is the geometric mean of the relative Mean Absolute Error (MAE) for each series:

$$\mathrm{AvgRelMAE} = \sqrt[n]{\prod_{j=1}^{n} \frac{\mathrm{MAE}_j}{\mathrm{MAE}_{j,b}}} \qquad (3\text{-}7)$$

$$\mathrm{MAE}_j\colon\ \frac{1}{h \times m} \sum_{i=1}^{m} \sum_{t=1}^{h} \left| y_{j,t} - \hat{y}_{j,t} \right| \qquad (3\text{-}8)$$

where $y_{j,t}$ is the observation at period t for series j, $\hat{y}_{j,t}$ is the forecast at period t for series j, n is the number of time series that we summarise accuracy over, MAE_j is the mean absolute error for time series j of the forecast over m origins and h-step ahead and $\mathrm{MAE}_{j,b}$ is the equivalent of a benchmark forecast. The Naïve forecast is considered as the benchmark, which simply extrapolates the last observation as a forecast for all h-periods. AvgRelMAE has desirable statistical properties as shown by Davydenko and Fildes (2013). This metric is simple to interpret. When the value is below 1 then the forecast is better than the benchmark and vice versa. We report the forecast accuracy of the competing methods using the AvgRelMAE, as in (3-9). We also report the classification accuracy of the meta-learners. The classification accuracy provides the percentage of cases that a min error forecast model is selected correctly by a selection methodology.

3.5 Results

3.5.1 Accuracy of base forecasters

The error metrics are reported for all time series as an overall, and for groups (Yearly, Quarterly, Monthly, and Other) separately. Table 3.5-1 reports the RelAvgMAE for the simple base forecasters on the 20% test set. The best performing forecast in each case is highlighted in bold. Unsurprisingly, the Theta model, which was the top performer in the M3 competition, is the most accurate. ETS(AAdN) model surpasses other ETS models overall, and the single exponential smoothing (ANN) results in lower error than the AAN and the MAM forecasts.

Table 3.5-1. AvgRelMAE of simple base forecasters

	AvgRelMAE					
	ANN	AAN	AAdN	MNM	MAM	Theta
Total	0.997	1.056	0.982	0.991	1.020	**0.865**
Yearly	1.002	1.031	1.018	1.005	1.054	**0.990**
Quarterly	0.985	0.994	0.962	0.981	0.988	**0.904**
Monthly	1.001	1.120	0.988	0.989	1.040	**0.840**
Other	0.997	0.850	0.845	0.997	0.846	**0.590**

Similarly, the accuracy of the complex base forecasters is reported in Table 3.5-2. In agreement with Athanasopoulos et al. (2017), THieF overall outperformed other forecasts. ForecTheta and MAPA achieved the best overall accuracy after THieF. On the other hand, Auto.Arima exhibited poor overall accuracy with an AvgRelMAE of over 1, which means a worse performance than naïve forecast.

Table 3.5-2. AvgRelMAE of complex base forecasters

	AvgRelMAE					
	ETS	Auto.Arima	MAPA	TBATS	ForecTheta	THieF
Total	0.937	1.009	0.903	0.945	0.902	**0.872**
Yearly	1.182	1.228	1.078	1.240	0.996	**0.962**
Quarterly	1.183	1.160	1.159	1.213	**1.138**	1.152
Monthly	0.790	0.928	0.775	**0.770**	0.817	0.773
Other	0.459	**0.440**	0.486	0.470	0.528	0.463

When we look at separate groups of time series, different forecasters performed best for each case.

Results from the simple and complex forecasters demonstrate that the latter were more accurate, as expected. It is of interest to observe how meta-learning can improve further on these results.

3.5.2 Meta-learner classification

The classification accuracy of competing meta-learners is reported in **Error! Reference source not found.**. The best result in each scenario (column) is highlighted in boldface, and the best overall results for the simple and complex forecasters are underlined.

Table 3.5-3. Classification accuracy of meta-learners using simple and complex base forecasters

	Simple		Complex	
Feature set	Tsfresh	Tsfeatures	Tsfresh	Tsfeatures
Decision Tree	18.14	18.97	26.29	25.79
Random Forest	**26.13**	**24.46**	**30.78**	**31.12**
XGBoost	24.21	20.13	29.45	27.79

Average	**22.82**	21.18	**28.84**	28.23

Random Forest and XGBoost models perform substantially better than Decision Tree meta-learner, which can be attributed to their inherent bagging and boosting, respectively. In both cases of simple and complex forecasters, the Random Forest showed superior performance to XGBoost.

The Tsfresh set of meta-features resulted in better accuracy on average for simple models, while the complex methods obtain better classification accuracy with Tsfeatures. However, Tsfresh has on average higher accuracy for both simple and complex models. We attribute this to the diverse pool of features included.

3.5.3 Accuracy of meta-learners

In this section, we compare the performance of the competing meta-learners. The results for the simple and complex forecasters are reported in Table 3.5-4 and Table 3.5-5, split by meta-learner and feature set. The best performance in each set of time series is highlighted in boldface.

Table 3.5-4. Meta-learning accuracy with simple base forecasters.

	AvgRelMAE					
	Decision Tree		Random Forest		XGBoost	
	Tsfresh	Tsfeatures	Tsfresh	Tsfeatures	Tsfresh	Tsfeatures
Total	0.962	1.028	**0.868**	0.928	0.885	0.939
Yearly	0.958	1.027	0.946	**0.896**	0.960	0.942
Quarterly	1.071	1.123	**0.931**	1.067	0.968	1.097
Monthly	0.979	1.031	0.863	0.939	**0.844**	0.945
Other	0.695	0.876	**0.638**	0.639	0.688	0.658

The Random Forest meta-learner demonstrated the best overall accuracy, in agreement with.

Comparing the best simple base forecasters' accuracy reported in Table 3.5-5 with the meta-learning approach, we can see that Theta (AvgRelAME: 0.865) performed slightly better than the best meta-learning approach (Random Forest with Tsfresh). As was reported in Table 3.5-5, the forecast accuracy of

Theta was far better than other simple base models, which can limit the advantage of the meta-learning approach.

The Tsfresh set of features performed better than the Tsfeatures set, in agreement with. In Table 3.5-5, we can observe a similar result for the complex forecasters, with the Random Forest meta-learner, relying on Tsfresh performing best.

Table 3.5-5. Meta-learning accuracy with complex base forecasters

	AvgRelMAE					
	Decision Tree		Random Forest		XGBoost	
	Tsfresh	Tsfeatures	Tsfresh	Tsfeatures	Tsfresh	Tsfeatures
Total	0.868	0.876	**0.824**	0.857	0.843	0.890
Yearly	1.025	1.034	**0.881**	1.028	0.985	1.143
Quarterly	1.101	1.067	1.084	1.070	1.044	1.072
Monthly	0.768	0.778	**0.737**	0.748	0.761	0.771
Other	0.475	0.498	0.475	0.467	**0.428**	0.470

By comparing the meta-learning results in Table 3.5-4 and the individual base forecaster performance in Table 3.5-5, we can confirm that in all cases, the meta-learning approach performed significantly better than the best base complex forecaster. Overall, THieF obtained an AvgRelMAE of 0.872, while the meta-learner achieved 0.824.

This result suggests some similarities of meta-learning with the forecast selection and combination literature. When there is a dominating forecast in the pool of potential models (or equivalently, very poorly performing forecasts) the selection and combination of forecasts is harmed, which can be resolved by first pre-filtering the forecasters considered in the pool (for a more detailed discussion see, Kourentzes et al. (2019b). For our case, Theta substantially outperformed all alternatives in the simple forecasters, and therefore the meta-learner only introduced learning errors. For the complex forecasters, all forecasts are competitive and perform well. This is also illustrated in the diverse best performer for each subset of time series reported in Table 3.5-5. Although the meta-learner can again introduce potential classification errors, it is now able to add value to the forecast selection by distinguishing between the various complex forecasters for each time series. Also note that all complex forecasters have some model selection or combination routine, which however is not able to accurately pick the best option for all-time series. The meta-learner using a much richer set of meta-features can complement this inherent mechanism in the complex forecasters, providing further accuracy improvements.

3.5.4 Feature importance in Meta-learning

We saw that the Random Forest meta-learner showed the best performance in both cases. Here, we explore further how the classifier used the meta-features. Although Tsfresh was overall better, it may be that it contains some meta-features that are not useful, or equivalently misses some of the beneficial ones in Tsfeatures. We extract the feature importance values for all individual meta-features and report the mean value of the relative importance for each group of features. The importance is calculated as in Hastie et al. (2009).

Table 3.5-6 reports the importance value of the 10 meta-feature groups for both simple and complex base forecasters. We can see that the Dickey-Fuller, Auto-Regressive, and Auto Correlation feature sets presented the highest importance values in both cases. Features based on time series quantiles and descriptive statistics showed the lowest importance value, suggesting that these two feature groups did not provide critical information for meta-learner to detect the best base forecaster for the time series.

We note that a considerable proportion of Tsfeatures meta-features were classed in the lower importance groups, explaining the accuracy results reported in the previous section.

Table 3.5-6. Importance value of feature groups

	Simple		Complex	
Feature group	rank	importance value	rank	importance value
Dickey Fuller	1	0.95	1	0.79
Auto Regressive	2	0.76	2	0.77
Auto Correlation	3	0.73	3	0.73
Energy	4	0.69	5	0.67

Entropy	5	0.68	7	0.65
Wavelet Transformation	6	0.67	4	0.68
Fourier Transformation	7	0.64	8	0.66
STL	8	0.60	6	0.64
Quantiles	9	0.57	9	0.58
Descriptive Statistics	10	0.41	10	0.44

To further explore the value of each group of meta-features, we construct Random Forest classifiers using initially all features (from both Tsfresh and Tsfeatures) and evaluate its accuracy. We then remove the least important group of meta-features and re-train the classifier and evaluate its forecasting accuracy. We repeat until we are left with only the most important group of features. This process is repeated for the simple and complex base forecasters separately. The results are reported in Table 3.5-7 and Figure 3-4.

Table 3.5-7. Comparison of the RelAvgMAE for simple and complex models

Number of feature groups used	1	2	3	4	5	6	7	8	9	10
Simple models	0.965	0.970	0.885	0.88	0.885	0.915	0.895	0.87	0.872	0.874
Complex models	0.853	0.840	0.86	0.83	0.815	0.805	0.815	0.816	0.82	0.825

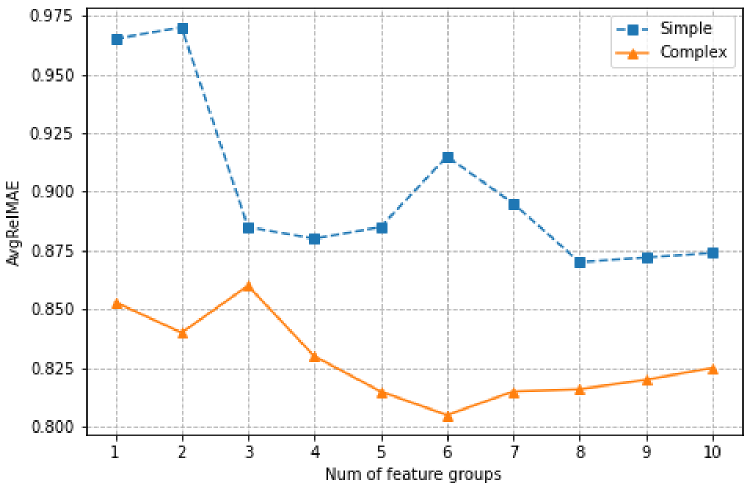


Figure 3-4. AvgRelMAE for cases containing different groups of features

In Figure 3-4, it is clear that removing feature groups can initially improve the forecast accuracy. When too many groups are eliminated the accuracy is harmed. The improvement appears to be more substantial for complex base forecasters, where the meta-learner adds most value. It can be observed that the meta-learner is more sensitive for the simple base forecasters, as limiting the features only makes learning more difficult. For both cases, we can see that adding all groups of features can decrease the performance of meta-learner (see Table 3.5-7), suggesting the suboptimality of purely judgementally generated features. This insight calls for additional research on algorithmically filtering features.

3.5.5 Model selection or model averaging

To prove the robustness of our result in a bigger data set and analyse the effect of model selection versus model averaging, in this experiment we randomly select 10000 of monthly M4 competition dataset as mentioned in 3.4.1. We begin our experiment by applying our presented forecasting models, Theta, Automated ETS (AutoETS), TBATS, AutoArima, and THieF. For the sake of simplicity, we used 4 meta-features from TsFresh package which shown their competence in the previous experiment:

- Auto Correlation

- Variation Coefficient
- Dickey Fuller
- Auto Regressive

The above features are used as the input of meta learner for selecting the suitable forecaster for each series.

For meta-learning, we used Random Forest for both selecting and averaging strategies. The meta-learner is able to produce probabilistic predictions for choosing the best forecasters. We used these probabilities to compute a weighted average of forecasts in the averaging scheme.

Table 3.5-8 summarises the performance of each model individually in addition to the result of meta-learning with the averaging and selection strategies on the M4 dataset to forecast test data for 10000 time series. As you can see, Meta-Learning shows a superior performance even using four meta-features (we use AutoETS error as a benchmark for all models). Both meta-learning approaches outperform the individual forecasters, while the averaging obtains better result than model selection.

Table 3.5-8. Overall forecasting performance for 12 steps ahead of monthly M4

Model	AvgRelMAE
Meta-Learning (Averaging)	0.9302
Meta-Learning (selection)	0.9458
ThieF	0.9812
AutoETS	1
Theta	1.0295
TBATS	1.1528
Auto Arima	1.231

It is worth checking how meta-learner uses our selected four features to choose the best forecasting model. As the meta-Learner is a standard classifier, we can use SHAP (Lundberg and Lee, 2017) or any other method to explain and interpret its prediction. In Figure 3-5, we analyse the most important features for deciding whether to use the AutoETS or not. From top to down, we plot the SHAP for the Auto Correlation, Variation Coefficient, Dickey Fuller, and Auto Regressive. It turns out that among the computed meta-features, Auto Correlation is the most important feature in selecting the AutoETS as the main forecasting model.

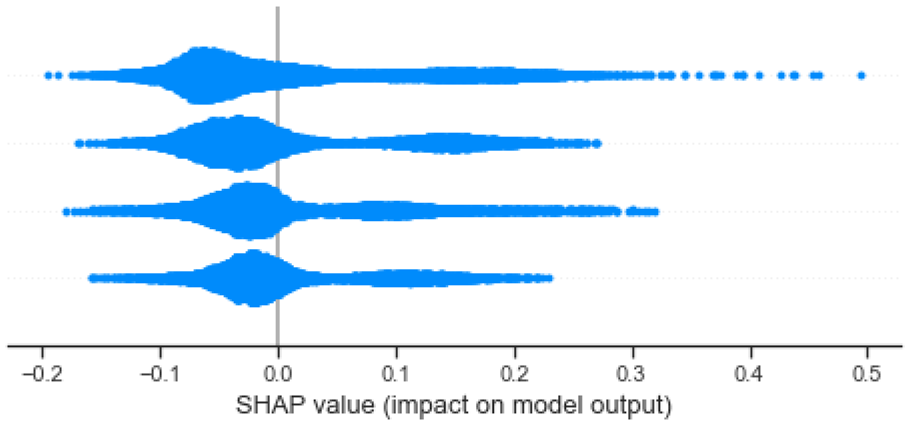


Figure 3-5. SHAP value for the meta-features importance

3.6 Data-driven meta-features

Though meta-learning approach have shown their superiority in terms of model selection and combination, the meta-feature extraction process is challenging and potentially unreliable in the context of big time series. According to Ma & Fildes (2020), current forecasting studies related to meta-learning rely heavily on manually selected meta-features. However, in order to extracting useful meta-features, business analysts are required to deeply understand all time series features and have rich experiences in feature selection, which is significantly complicated and challenging for them. These requirements make it difficult for meta-learning approach to be widely used in real-life business time series forecasting.

From a generative point of view, model selection is equivalent to discovering the underlying data generation process. From this perspective, the forecasting model that is able to simulate the data generation process is the ideal choice for forecasting. Therefore, we propose a deep unsupervised learning based on Autoencoder to extract rich features from the latent layer of the data regeneration process. Here, we want to analyse the potential of this new group of deep meta-features in comparison with the manually calculated features from the meta-learning packages.

As an example, consider a set of time series sampled from two different variants of the exponential smoothing family say ANN and MMM with random parameters. In order to compare our proposed meta-features with statistical meta-features, we first extract a set of statistical features using TsFresh containing 426 features for each series. The Figure 3-6 depicts the feature space after dimensionality reduction.

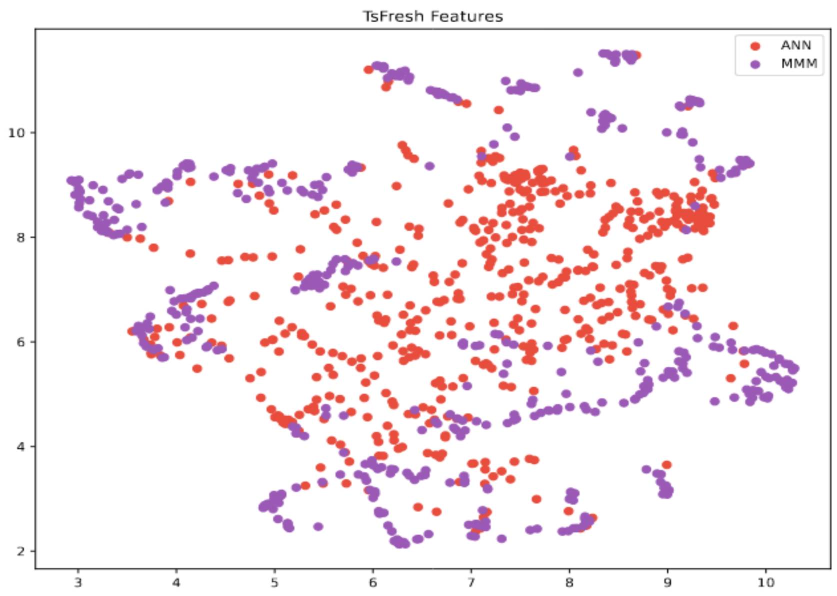


Figure 3-6. classifying the two variants of the exponential smoothing family without using Auto Encoder

As you can see in the Figure 3-6, classifying the two variants of the exponential smoothing family is not trivial in this space. The reason is that the statistical and hand-crafted features are not necessarily related to the data generation process. On the other hand, since the objective of an Auto Encoder is to minimize the reconstruction loss, the latent space has to contain the information that is enough to re-generate the data. More specifically, the extracted features should be rich enough for mimicking the data generation process. Therefore, it is not surprising to see that the features extracted from Auto Encoder could be more informative, especially for model selection.

In the next step, we extract a set of meta-features for the same data using an MLP Auto Encoder [See 3.3.4]. As you can see in the Figure 3-7, in this space two classes can be separated easily using a linear classifier.

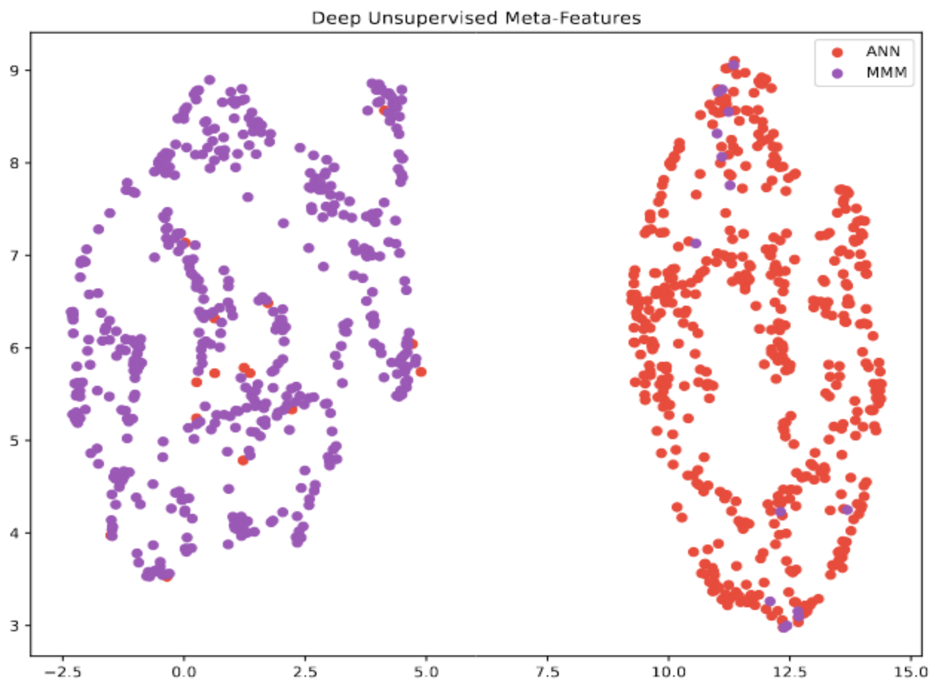


Figure 3-7. classifying the two variants of the exponential smoothing family with using Auto Encoder

Note in this example, the feature vector for each series extracted using TsFresh had a dimension of 426 while the dimension of Auto Encoder's latent space was 8 which is more efficient for any analysis including model selection.

3.7 Limitations

Since we are faced with solving the classification problem in meta learning, the limitations that exist in classification problems can also be discussed here. One of the most critical limitations in classification problems is imbalanced data that cause bias. Training a machine learning model on an imbalanced dataset can introduce unique challenges to the learning problem. Imbalanced data typically refers to a classification problem where the number of observations per class is not equally distributed; often, you'll have a large number of data/observations for one class (referred to as the majority class), and much fewer observations for one or more other classes (referred to as the minority classes)

In meta-learning, the imbalanced data problem arises when one of the algorithms performs better than the other. To avoid this situation, we need to use more accurate base forecasters. On the other hand, machine learning forecasting algorithms usually have better performance, so we should not use only one of them.

If the nature of the data is such that most time series are in one class, and it is impossible to avoid imbalanced data, we need to use the methods to solve this problem. These methods help the meta-learner and prevent bias. They concentrate on modifying the training set to make it suitable for a standard learning algorithm. This can be done by balancing the distributions of the dataset which can be categorized in two ways (Thabtah et al. 2020):

- **Under-sampling**

The primary under-sampling technique that arbitrarily eliminates examples of the majority class to balance the dataset is known as random under-sampling (Kotsiantis, Kanellopoulos, & Pintelas, 2006). Researchers such as Guo et al. (2008) reveal the primary drawback associated with this method to be the disposal of useful information that can prove to be crucial in the later classification stages. Many under-sampling approaches have been proposed such as Condensed Nearest Neighbour Rule (CNN), Wilsons Edited Nearest Neighbour Rule (ENN), Neighbourhood Cleaning Rule (NCL), Tomek Links, and One-sided selection (OSS) (Hart, 1968; Tomek, 1976; Wilson, 1972; Laurikkala, 2001).

- **Oversampling**

Oversampling is another common sampling approach used to deal with an imbalanced class problem. Various oversampling strategies are available including random oversampling, focused oversampling, and synthetic sampling (Estabrooks, Jo, & Japkowicz, 2004; Chawla et al., 2002; Kubat & Matwin, 1997). The method in which the instances of the minority class are randomly replicated until they have equal representation is known as random oversampling (Buda, 2017). Random oversampling has two major shortcomings: it increases the possibility of overfitting of the classifier on the training dataset, and if the original data already has high dimensionality, it mounts the computation cost thus increasing the training time of the classifier (Chawla, 2005). In focused oversampling, only those minority class values with samples occurring on the boundary between the majority and minority class values are resampled. However, Chawla et al. (2002) state that these methods of oversampling by replication lead to a more specific decision region of the minority class. To overcome this and broaden this decision region, they propose an advanced heuristic oversampling technique called the Synthetic Minority Oversampling Technique (SMOTE). SMOTE is a technique in which oversampling of the minority class is carried out by generating synthetic examples. These new synthetic minority class examples are the result of interpolation between closely located minority class samples. Chawla et al., (2002) describe the process of SMOTE as calculating the nearest same-class neighbours for every minority example and then based on the required oversampling rate, randomly choosing from these neighbouring examples. The synthetic examples are then generated at random points along the line segments joining the minority examples with these chosen neighbours. This process expands the decision region pertaining to the minority class.

3.8 Conclusion

In this study, we investigate the design of meta-learning for time series forecasting. To explore the impact of different elements of meta-learning, including the meta-learner, the meta-features, and the pool of base forecasters, we studied three tree-based classifiers as meta-learners, three distinct sets of features, namely the Tsfresh, the Tsfeatures, and a new deep data-driven features from MetaTs, two groups of simple and complex base forecasters grouped according to having some internal model selection and/or combination, or not. Finally, model selection and model averaging in meta-learning is analysed.

We find that the Random Forest and XGBoost classifiers performed best. Although the result is not unexpected, it is interesting as these classifiers arguably do not require careful feature selection.

We find that the larger Tsfresh set of meta-features performs best, compared to Tsfeatures. However, when we looked at the various groups of meta-features included in these sets, we showed that by eliminating the least important features, we could increase accuracy further. From an aggregated viewpoint we identified the groups of meta-features that used model characteristics (e.g., based on the statistical test, autocorrelation, etc.) to be more informative than descriptive features (such as quantiles, mean, etc.). This calls for more research into algorithmically choosing the appropriate meta-features from a large pool of features. We note that even though the best performing Random Forest does not require fine-tuning of its inputs when groups of meta-features were eliminated the forecast accuracy of the meta-learner improved. Naturally, excessive elimination of features was damaging.

We investigate the effect of feature selection on meta-learning performance. We show that: first, increasing the number of features does not necessarily improve meta-learning performance; it may even have an adverse effect. Second, the quality of meta-features is very important in the performance of meta-learning, and features that are data-driven, such as rich features extracted by autoencoders, have more quality than pre-defined features. They give better feature space segregation for enhancing the classification power and consequently model selection performance.

In terms of the simple and complex base forecasters, we show that using more accurate base forecasters can help the meta-learner. We argue that when a base forecaster dominates over all other forecasters, then meta-learning does not add value and can potentially even harm the forecast accuracy, by introducing learning errors. On the other hand, when high-quality base forecasters are used, meta-learning can provide even more benefits.

Finally, we improve the forecast accuracy by model averaging the time series; note that the selection of an appropriate pool of forecasters, ensuring both quality and diversity, appears to be essential for the good accuracy of the meta-learner.

4 Deep Learning for Forecasting Model Selection

4.1 Introduction

Selecting the appropriate forecasting model among a wide variety of alternatives remains a challenging task. The modeller has to identify a model that adequately captures the key features of the unknown underlying data generating process, while at the same time managing the complexity of the model to ensure good forecasting performance. Therefore, building a decision system that can select the best forecasting algorithm between candidates remains a worthwhile endeavour.

There are multiple approaches to addressing this problem in the forecasting model selection literature. Typically, this has been done by comparing individual approaches, such as information criteria methods and empirical cross-validated errors. The information criteria approach relies on using penalised likelihood functions for evaluating the goodness of fit, e.g., Akaike information criteria (AIC) and Bayesian information criteria (BIC) and their variations (Burnham and Anderson, 2002). In contrast, cross-validation relies on using a validation sample to evaluate the model's performance (Fildes and Petropoulos, 2015). Although we reduce the available sample for model fitting, the validation sample enables us to assess multiple-step-ahead forecasts and different loss functions, while manipulating the time series with transformations, or changing the sample size, do not invalidate the comparisons of forecasters performance.

Individual model selection has to implement all candidate models to evaluate their performance and select between them. This wrapper procedure is computationally intensive and often infeasible to implement sufficiently due to the size of the forecasting problem at hand. For example, in modern retailing applications, we often need to forecast repeatedly several thousand of time series at a high frequency (Fildes et al., 2019). On the contrary, filter approaches rely on identifying the correct candidate model, without implementing all alternatives and evaluating them, exemplified in the methodology of ARMA model building (Box et al., 2015). Meta-learning, a filter methodology, is a learning approach which selects the best forecasting model based on characteristics (meta-features) of a time series, without the implementation of the candidate forecasts. Therefore, this can have substantially less computational costs for the application. Although a filter approach such as meta-learning can select the appropriate model without implementing all models in the data set, its performance relies strongly on the appropriateness of the extracted features from the time series (meta-features).

So far, existing research in meta-learning focuses on extracting a large collection of universal meta-features from time series and using a classifier (meta-learner) to map the connection between meta-features and model selection. For example, Talagala et al. (2018a) proposed a meta-learning framework, called FFORMS (Feature-based FORecast Model Selection), that uses a set of 45 meta-features and a Random Forest as meta-learner to select the best single model from nine base forecasters. The performance of meta-learning is evident in the recent M4 competition (Makridakis et al., 2018) with Montero-Manso et al. (2020) by proposing FFORMA (Feature-based FORecast Model Averaging) that ranked second.

The proposed meta-learning methods depend on meta-features that are constructed using expert prior knowledge and judgment. Furthermore, there is no evidence that calculating a set of prior designed features can capture all intrinsic properties embedded in data. Thus, in this research, inspired by deep feature learning for image classification (Bengio et al., 2013, LeCun and Bengio, 1995, LeCun et al., 2010), we design a meta-learning framework based on deep convolutional networks that can learn a feature representation from raw time series automatically, without any intervention from the modeller. Among the many deep learning algorithms available, convolutional neural networks (CNN) have achieved a breakthrough accuracy in general pattern recognition tasks, such as image classification by automatically learning features from the dataset (Simonyan and Zisserman, 2014, Van den Oord et al., 2013), and sequential data classification, language modelling and other related problems in natural language processing (Collobert and Weston, 2008, Kalchbrenner et al., 2014).

This chapter evaluates the potential of deep learning for time series model selection, without relying on external features. The time series model selection can be treated as a one-dimensional recognition task, and the proposed CNN allows learning different levels of representations (the meta-features) together with a classifier, jointly and automatically. We provide empirical evidence of the efficacy of the approach against widely accepted forecast selection methods and discuss the advantages and limitations of the proposed alternative. While the majority of time series meta-learning have focused on meta-features, in this chapter, we focus on abstracting these directly from the time series, providing a comparable approach in inputs and complexity to conventional time series selection approaches. We evaluate the robustness of the proposed approach for both long and short time series, using simulated and real data, to better highlight the conditions when the proposed approach performs well.

The structure of this chapter is as follows: Section 4.2 presents a background in model selection approaches and deep learning studies for time series. Section 4.3 explains the proposed methodology. Section 4.4 describes the experimental setup and summarises our findings. Section 4.5 presents a discussion of the approach, and finally, Section 4.6 concludes and indicates further research.

4.2 Background of model selection in time series

4.2.1 Conventional forecast selection and combination

4.2.1.1 Selection

There have been many studies elaborating on forecasting model selection. Traditionally, model selection was the outcome of a detailed time series exploration, where the analyst would identify the influential patterns in a time series, such as seasonality or the presence of unit roots, among other features such as outlying observations, to identify the best forecasting model (Box et al., 2015, Makridakis et al., 2008). This exploration approach was given further credence by Petropoulos et al. (2018), showing that even non-experts could perform accurate model selection when following a structured exploration approach. However, this approach is very subjective and often very difficult to automate and scale up (Ord et al., 2017). Nonetheless, it has resulted in the development of a multitude of statistical tests that can support automatic model building (for example, see(see Hyndman and Khandakar, 2007), but also the development of expert systems and rule-based forecasting approaches, where ad-hoc procedures are used to identify the appropriate forecasting method (Goodrich, 1992, Adya et al., 2001).

A more theoretically consistent approach to model selection has been to rely on information criteria. The use of information criteria to choose between forecasting models has been central in statistical and econometric model building. Information criteria, such as AIC or BIC, penalise the optimised likelihood function for the model’s complexity, so as to avoid overly flexible models that may lead to overfitting and poor forecasting performance (Burnham and Anderson, 2002). Although the different information criteria have different foundations and properties, in terms of forecasting performance they often lead to similar results, even if a different model is selected (Billah et al., 2006). Furthermore, they have been shown to reliably identify the appropriate model form for many popular forecasting model families with no user intervention (Burnham and Anderson, 2002, Hyndman and Khandakar, 2007, Hyndman et al., 2008). However, information criteria require that the likelihood between the alternative models is comparable, which excludes comparing models with different data transformations, or differencing, and requires care in the construction of lagged realisations. More crucially they require the existence of a statistical model to underly the forecasts, and therefore cannot be used with many popular heuristic forecasting methods, machine learning or artificial intelligence. This ultimately makes then inappropriate when diverse forecast need to be compared.

To overcome the limitations of information criteria, the modeller can rely on cross-validation, which has been applied in various flavours in time series forecasting (e.g., (Fildes and Petropoulos, 2015); (Kourentzes et al., 2019a). Cross-validation does not rely on the existence of a statistical model but rather compares the performance of competing forecasting approaches on a validation sample. As some sample needs to be retained for the comparison, cross-validation exchanges flexibility with loss of training sample. In contrast to information criteria, there is no need for the existence of a comparable optimised likelihood, which has made cross-validation the method of choice for many machine learning approaches, as a likelihood function is often not available (Ord et al., 2017). Stone (1977) showed that AIC is equivalent to the one-step-ahead out-of-sample cross-validated error. However, as cross-validation is more flexible, both the forecast horizon and the loss function can be adjusted to fit better to the relevant forecasting problem (Fildes and Petropoulos, 2015). Nonetheless, the need for an adequate sample is a significant weakness of the approach, as on the one hand, many applications have limited sample availability, and on the other hand, cross-validation on a limited sample is unreliable (Fildes and Petropoulos, 2015). In practice, this forces the modeller to often rely on the methodologies above.

All the above approaches, either using information criteria or cross-validated errors require the deployment of the competing forecasting algorithms, increasing the computational cost of the model selection substantially. This is a common feature of ‘wrapper’ approaches that need trailing of all alternative models (Barak et al., 2015). As with time series exploration and statistical testing, ‘filter’ approaches attempt to indicate the appropriate model without the need to deploy all alternatives.

To this end, meta-learning is a promising alternative, where the so called ‘meta-features’ are extracted from a time series and used as inputs to a classifier to select the best forecasting model. Meta-learning overcomes the limitations of information criteria in requiring a comparable optimised likelihood, but also does not require a dedicated sample for the calculation of the cross-validated errors. Furthermore, meta-learning is typically implemented as a filter and does not require deploying the competing models first, avoiding the related computational costs. Using model selection with meta learning in forecasting was first proposed by Arinze (1994). Chu and Widjaja (1994) proposed a neural network system to select among several exponential smoothing models using the autocorrelation function. Since then, many attempts have been made to improve the results of these filter recommender systems for time series analysis. Lemke and Gabrys (2010a) implemented 15 simple and combined forecast models, and used a support vector machine, neural network, and decision tree as meta-learning models in NN5 time series forecasting. Zhou et al. (2012) proposed a rate-based meta-learning model to identify the most suitable back-propagation neural network (BPNN) models in forecasting the gold price. Matijaš et al. (2013) applied the meta-learning approach to multivariate time series of four load forecasting tasks. Kück et al. (2016a) studied the use of different feature sets for a neural network meta-learner, and Talagala et al. (2018a) exploring further potential of meta-features. Ma and Fildes (2020) proposed a meta-learning framework based on convolutional neural networks, which uses external meta-features, but also the raw time series to obtain combination weights for a pool of base-forecasting methods, demonstrating very promising performance.

Although the choice of a classifier is important, creating a representative set of meta-features has been a focal point for the meta-learning community, with open questions on both designing and selecting useful meta-features, given the time series and the competing forecast considered. Overcoming this has been one of the central motivations for this work.

4.2.1.2 Combination

An alternative to selecting a single forecast is to combine different forecasts, and its benefits are widely accepted (Elliott and Timmermann, 2016). Model combination leads to a reduction of model uncertainty, since the modeller does not have to choose a single best model, simplifying the forecasting process, as well as mitigating the forecast error variance. Chan and Pauwels (2018) demonstrate that simply selecting a single model on cross-validated errors will lead to a biased selection, supporting a combination approach.

In the prediction literature, model combination has received a lot of attention (Barak et al., 2017). The combination of predictions has resulted in many superior algorithms, such as the Random Forests, which is a combination of multiple decision trees for classification and regression problems (Breiman, 2001), bagging of time series to improve the performance of exponential smoothing (Bergmeir et al., 2016), combining forecasts from different temporally aggregated versions of the data (Kourentzes et al., 2014), forecast islands, a heuristic to automatically identify forecast pools for combination (Kourentzes et al., 2019b) among others. Montero-Manso et al. (2020) proposed a feature-based forecast model averaging, which is a weighted combination of 9 forecasting models using XGboost as a meta-learner and obtains the second rank in the M4 forecasting competition. Ma and Fildes (2020) explore a similar approach with CNN.

The combination literature has faced the paradoxical finding that simple combination methods, such a simple average or median, often perform better than theoretically elegant approaches that rely on optimal weighting schemes, for a variety of optimality criteria. Smith and Wallis (2009) provide an explanation that this is due to estimation errors of the combination weights, which propagate to the combined forecast. Therefore, fixed suboptimal weights can perform better than estimated weights. This has led to various innovations in the combination literature, which tries to restrict the estimation uncertainty, with forecast pooling being one such approach that first eliminates forecasts from the combination, before estimating weights for the remaining ones (Kourentzes et al., 2019a), thus limiting the combination weight uncertainty. Kolassa (2011) showed that combining forecasting models using AIC weights leads to more accurate results than selecting a single model on AIC. A distinctive difference between AIC weights and conventional optimal combination weights is that the former is calculated indirectly. Given a model we calculate its weight relying on its AIC, without performing any further estimations. This limits the uncertainty that propagates to the combined forecast. We argue that combinations of forecasts using meta-learning have similar properties, as any weights are conditional on the (pre-calculated) meta-features and therefore are indirectly calculated.

4.2.2 Deep learning approaches for time series forecast selection

Convolutional neural network (CNN) is a deep learning method, where different layers with specialized architecture (LeCun et al., 1998). CNN consists of three main layers (i) the convolution layer; (ii) the pooling layer; and (iii) a fully connected layer. Each CNN has two steps for training the network, which is feedforward and back-propagation. In the first step, the input vector feeds to the network, and the output calculated. The network errors are computed by output results and applied to regulate the network parameters or, in a better word, the training process. Herein, the output of the network is compared with the actual response by a loss function, and the error rate is estimated. In the next step, based on the estimated error rate, the learning of the back-propagation process is continued until a predetermined bound reached.

There are several CNN applications on time series with the main focus being on time series classification (Wang et al., 2016, Cui et al., 2016b, Zhao et al., 2017). There is limited research on using CNN for time series forecasting. To the best of our knowledge, the first application of CNN for time series modelling is for generating raw audio waveforms with the Wavenet network (Oord et al., 2016). This model, when applied to text-to-speech, yields state-of-the-art performance. Recently, Li et al. (2020) transferred time series images into recurrence images, then extracted features with a max-pooling layer, which are then used for forecast model averaging. However, analyzing time-series images results in sparse matrixes which distort the functionality of pooling layers. Finally, Salinas et al. (2020) proposed DeepAR, a probabilistic forecasting methodology based on training an autoregressive recurrent neural network model on a large number of related time series.

4.3 A convolutional neural network for meta-learning

A CNN is made up of different components, which when working together give it its properties. Here we will first introduce the various components, and then show how these are used for our purpose, while attempting to make connections with conventional model selection procedures.

We restrict the description of the various components to the 1-dimensional case, which is apt for univariate time series data, rather than the higher-dimensional cases, which are more common in areas such as image recognition. For a detailed description of CNN, we refer the reader to Goodfellow et al. (2016).

4.3.1 The convolution layers

Given a time series $\boldsymbol{y} = y_1, \dots, y_n$, where n is the sample size, the CNN applies a series of the so-called *filters* (or *kernels*) to highlight different aspects of the data. Let us focus on the operation of a single filter, which itself requires a one-dimensional vector weight $\boldsymbol{w}$, containing f elements (commonly referred to as the receptive field), that is multiplied with the local f observations of $\boldsymbol{y}$:

$$\boldsymbol{o_t} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{f}} \boldsymbol{w_i y_{t-\left\lfloor \frac{f}{2} \right\rfloor + i - 1}} \tag{4-1}$$

where $\boldsymbol{o_t}$ is the value of the filter at time t that is computed by the vector multiplication of $\boldsymbol{w}$ and the $\boldsymbol{f}$ observations symmetrically surrounding $\boldsymbol{y_t}$. That already suggests that for the one-dimensional convolution $\boldsymbol{f}$ conveniently takes odd values and that the $\boldsymbol{o_t}$ cannot be computed for the first and last $\lfloor \boldsymbol{f/2} \rfloor$ observations. Observe that the filter is very similar to the well-known centered moving average (Ord et al., 2017)**,** but with weights $\boldsymbol{w}$. Different weights highlight different aspects of the time series. For example, if the weights are approximately equal, this will filter high-frequency components of the time series. We can also draw parallels to conventional time series transfer functions and autoregressive models, however, it is important to note that for convolution filters there are no restrictions on the weights to satisfy the usual invertibility and stability conditions (Box et al., 2015), given that they do not perform a forecasting task.

Within a convolution layer of the CNN, each filter is embedded into a neuron:

$$z_t = g(o_t + b), \tag{4-2}$$

where $g(\cdot)$ is a nonlinear activation function, typically for deep learning being the Rectified Linear $\max(0, x)$, with x being the input, and b is a constant, referred to as the bias. Additionally, we may consider a stride parameter that instructs the filter to be applied to every observation. The stride parameter is typically 1 (every observation) or 2 (every second observation). The stride helps to reduce the computational cost by down-sampling the input (Krizhevsky et al., 2012).

Within a convolutional layer, we can have k such filters, each with each its own set of weights and bias. However, to simplify the design of the layer, it is common practice to use the same number of elements, activation functions, and stride across all filters. Therefore, given a time series $\boldsymbol{y}$ the output of the convolutional layer is a matrix with dimensions $k \times (n - \left\lfloor \frac{f}{2} \right\rfloor)$ and requires $k(f + 1)$ parameters to be trained.

In a well-trained network, each of the filters will highlight a different feature of the time series. Herein lies a key argument for our approach. Instead of relying on externally designed and selected features, we let the CNN identify the most appropriate ones for the given dataset and available forecasts to choose from for the meta-learner.

4.3.2 The pooling layers

To mitigate overfitting and to reduce the computational complexity, we use a pooling layer, which simply aggregates inputs locally. In the pooling layer, we process each input (the output of a single filter from the convolution layer) independently. There are two widely used pooling operators, the maximum and the average. Using these, we calculate either the maximum or the average every s observations in a non-overlapping manner. Pooling effectively operates as a down-sampling mechanism.

In CNN, it is common to follow a convolutional layer with a pooling layer. For example, suppose we have a time series with 100 observations that is inputted in a convolution layer with 10 filters and a receptive field of 5 observations. The output of the layer will be a matrix of 10 rows with 96 observations each (we lose $\lfloor 5/2 \rfloor$ observations from the beginning and the end of the time series due to the convolution). This can then be fed to a pooling layer, for example, with a stride $s = 2$. Every s observations are aggregated into a single observation (in a non-overlapping fashion), reducing the data to 10 rows of 48 observations. In a deep convolutional network, the output of the pooling layer will typically be inputted into a subsequent convolution layer, and so on. Observe that the subsequent layers, on the one hand, will require fewer parameters, and on the other hand, will use data with higher abstraction, due to the filtering and the pooling. After multiple steps of convolution and pooling, the CNN internally represents highly informative meta-features of the input time series, without the intervention of the modeller.

The (one dimensional) pooling layer has a well-known time series equivalent, which is the non-overlapping temporal aggregation. For instance, Kourentzes et al. (2014) use the average as a temporal aggregation operator to filter high-frequency time series components, such as noise and seasonality, to help highlight different features in the data. It should be noted that there is a trend in CNN networks to replace the pooling layer with a stride in the convolution layer of equal size (Springenberg et al., 2014). This mirrors the use of flow and stock variables in time series temporal aggregation, where in the first, we aggregate observations throughout the period of interest, while in the latter, we measure only at intervals of the period of interest.

Once we consider the multiple iterations of convolution and pooling layers in a deep convolutional network, there is a further analogy with time series models from the literature. Kourentzes et al. (2014) and Athanasopoulos et al. (2017) propose using multiple temporal aggregation views of a time series to mitigate modelling uncertainty, by purposefully highlighting different time series components, and demonstrate that this approach leads to increased accuracy over selecting a time series model using only on the original data. Kourentzes et al. (2017) provided evidence that it is the multiple levels of temporal aggregation that help gain the accuracy improvements, rather than the act of aggregating itself, demonstrating that considering the higher abstractions of the original time series data can help in forecasting. CNN performs the analogous data transformation internally, with one additional advantage. In the multiple temporal aggregation literature, the temporal aggregation always assumes equal weights over the observations within the aggregation span, while in CNN, the filters in the convolutional layer allow not only weighted representations but also multiple ones, as it is common to use multiple filters. Therefore, we argue that this is another potential strength of CNN over conventional time series modelling approaches, in that the CNN can learn the appropriate representations from the data, rather than relying on external modelling decisions.

4.3.3 The classifier

Once we have processed the time series through multiple convolutional and pooling layers, we have a set of highly abstracted rich meta-features. These are inputted into a fully connected layer, much akin to a layer from a conventional neural network classifier, that has the task to classify the time series to a forecaster, given the constructed meta-features. The output of this layer indicates what forecaster should be used for the given time series. This resembles a conventional meta-learner in its purpose and structure. Figure 4-1 summarizes the different elements of the proposed meta-learner CNN.

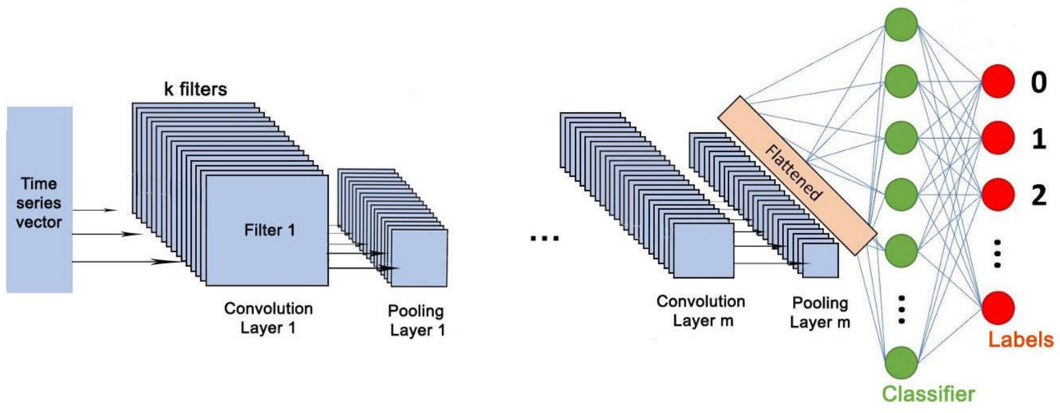


Figure 4-1. Convolutional network structure

The key difference to conventional meta-learners lies in the training of the network. Typically, we use a set of pre-defined external features that are inputted into a classifier that acts as the meta-learner. Therefore, in learning the associations between time series and forecasts, the classifier is inherently restricted given its inputs. This is in contrast with the CNN, where both the construction of the meta-features and the classification are learned simultaneously. This provides increased flexibility to the classification capabilities of CNN, as the constructed meta-features can be learned to better match the available pool of forecasts and the characteristics of the time series dataset.

We argue that this is yet another advantage of our proposed approach, as the CNN meta-learner is not biased in any way by prior choices by the modeller or the difficulty of generating appropriate external meta-features. This difficulty lies not only in potential computational challenges but also in the various tools and methods we have available to characterize time series, which may or may not help the classifier match the time series to a given pool of forecasts. To exemplify this argument, different time series model families can process information differently. For instance, the exponential smoothing family of models considers time series as the combination of three components, the local level, the local trend, and the local seasonality, interacting in an additive or multiplicative way, between them and with the innovation term (Gardner Jr, 2006). In identifying the best exponential smoothing model, exploration tools such as the autocorrelation function, or the partial autocorrelation function are of little use. Furthermore, standard unit root tests are also of little use, as all exponential smoothing models contain a unit root, even though we classify them (perhaps unhelpfully) into level and trend models.

On the other hand, Petropoulos et al. (2018) demonstrated that classical time series decomposition provides useful information in distinguishing between exponential smoothing models. When considering ARIMA models, we can arguably make the opposite arguments, where the autocorrelation and partial autocorrelation functions are useful, as are unit root tests (Box et al., 2015), whereas classical time series decomposition is of limited use. This illustrates that different model families may rely on different sets of meta-features, which remains an open question in the literature. The proposed CNN approach overcomes this by matching the learned meta-features with the pool of forecasts to assign to each time series.

4.3.4 Forecast selection and combination

The proposed CNN can be used for either selecting a forecast or providing combination weights for the considered forecasts. The classifier relies on the softmax function, which outputs for each of the potential forecasts a normalized probability that it is the appropriate one. Suppose we are considering m alternative forecasts to choose from. The final element of the CNN is a softmax function that expects a m-dimensional vector $\boldsymbol{q}$, from the classifier layer and outputs a m-dimensional vector $\boldsymbol{\sigma}$ of normalized probabilities:

$$\sigma_i = \frac{e^{q_i}}{\sum_{j=1}^{m} e^{q_j}} \tag{4-3}$$

When the task involves the selection of a single forecast, we simply choose the forecast with the maximum probability σ_i. When the task involves the combination of the forecasts, we use the normalized probabilities as combination weights for the forecasts.

4.4 Experimental Evaluation

In this section, we outline the problem and data set as well as the experimental setup that we use to empirically evaluate CNN against conventional forecast selection and combination methodologies.

4.4.1 Datasets

We generate simulated time series from known data generation processes, to assess the ability of the proposed CNN to identify the correct model. For this purpose, we rely on the exponential smoothing (ETS) families of models, which has been shown to perform well for business time series from several applications (Makridakis and Hibon, 2000, Gardner Jr, 2006). Furthermore, exponential smoothing remains one of the most widely used models in business practice (Weller and Crone, 2012), and therefore being able to select the appropriate ETS model form is very relevant to practice. In a survey of forecasting practices, the exponential smoothing family of models is used in almost 1/3 of times (32.1%), with averages coming second (28.1%) and naïve methods third (15.4%). More advanced forecasting techniques are only used in 10% of cases.

The ETS family of models includes possible combinations of trend, season, and innovations terms, resulting into 30 distinct models. Table 4.4-1 provides the 15 additive error models, which are mirrored by an equal number of 15 multiplicative error variants (Hyndman et al., 2008). We consider two model selection scenarios: (i) a 4-model set of alternatives; and (ii) a 15-model. The 4-model option includes the ANN, ANA, AAN, and AAA cases from Table 4.4-1, capturing four relatively easy to distinguish cases. The 15-model scenario considers all alternatives in Table 4.4-1, constituting a much more challenging model selection problem, as many options can appear almost identical for certain parameters ranges (Hyndman et al., 2008). For each scenario, we consider a ‘short’ and ‘long’ history case, with 48 and 144 monthly observations respectively. For each combination of scenario, sample, and model, we generate 10,000 time series. Therefore, for the 4-model case, we have 40,000 series, and 150,000 series for the 15-model case for each history length. All series are generated randomly using the smooth package (Svetunkov, 2017) for R (Team, 2013). We simulate these time series with random number generation function and the error term are randomly chosen between Normal, T-student, Uniform, and Beta distributions. Parameters for each distribution are selected randomly but mainly around the following ranges: for Beta: sshape1=1.5, sshape2=1.5; for Normal: mean=0, SD=100; for Uniform: min=-0.5, max=0.5; and for T-student: mean=0, SD=100). The frequency of generated time series is equal to 12.

Table 4.4-1. Additive error ETS models

Trend Component	**Seasonal Component**		
	N (None)	A (Additive)	M (Multiplicative)
N (None)	ANN	ANA	ANM
A (Additive)	AAN	AAA	AAM

A_d (Additive damped)	AA_dN	AA_dA	AA_dM
M (Multiplicative)	AMN	AMA	AMM
M_d (Multiplicative damped)	AM_dN	AM_dA	AM_dM

To facilitate the training requirements of the proposed CNN, we split the data into three sets, with 80% of the time series for training, and the remaining two 10% sets for validation and test set purpose. We train the network to assign the appropriate forecasting model in the training set series. Then it is used to label the series belonging to the validation and tests sets.

In general, CNN needs a very large sample for training. This is also true for our proposed CNN meta-learner. This limits its applicability for many business forecasting applications. One way to overcome this limitation is to rely on transfer learning, where the network is trained on a large, often synthetic, dataset, and then applied to a different one with no further training. To assess the efficacy of this, we use the M3 competition dataset of monthly time series (Makridakis and Hibon, 2000), that contains real time series from various sources. These series exhibit different combinations of trend and seasonality, as well as having periods of outlying observations. We use the trained network of both the 4-model and the 15-model CNN for long time series, to access how it compares with conventional model selection done on each series of the M3 dataset independently, as is the standard practice.

4.4.2 Convolutional Neural Networks Architecture

We implement the CNN in Python using the Keras framework (Chollet, 2015), using three convolutional layers with Rectified Linear unit activation functions (ReLU), with 100 filters and a receptive field of 12 periods. Each convolutional layer is paired with a pooling layer, using the average operator, and the stride is set to two. The receptive field of 12 periods, matches the sampling frequency of the data. Therefore, the filters, depending on their weights, are able to handle the seasonal nature of the time series. Too small receptive field can make the network struggle to represent the typical features of the time series, while larger values can reduce the available sample by removing the information at the edge of the series, which can be a limiting factor for the short time series (Zhao et al., 2017). We pair the convolutional layers with pooling layers, using a stride of 2. In total we employ 3 such pairs.

Moreover, in between the convolutional and pooling layer of the 2nd pair, we introduce a dropout layer. This helps to mitigate overfitting during training, by substituting a random 20% of the values passed from pair of layers to the next with 0. The use of dropout layers has been found to be an effective way to increase the generality of CNN and largely prevents overfitting (Goodfellow et al., 2016).

The output of the last convolutional - pooling layers chain is 2-dimensional (pooled observations × number of filters). We transform this into a column vector and input it into the last fully connected layer that is tasked with the classification. For this layer, we use the softmax activation function. The output is either a 4-model or 15-model probability vector, which we can use to either select the most probable model or combine them using these probabilities as weights. This structure is chosen by evaluating different options and hyperparameters on the validation set. We choose the one that obtains the highest classification accuracy.

We train the network using the rmsprop optimizer (Tieleman and Hinton, 2012) measuring the classification discrepancy between the selection of the CNN and the true model that was used to generate the time series in the training set. During training, all weights corresponding to the filters in the convolutional layers and the fully connected classification layer are updated. Therefore, during training, CNN learns both how to construct the most helpful meta-features, but also how to use them to classify the time series. We argue that this flexibility is a major difference for convolutional meta-learning, where these two steps are performed separately and arguably with a different loss function.

In choosing the hyper parameters and activation functions of CNN models, we use KerasTuner as a robust hyper parameter optimisation framework and its random search module. Finding the optimal neurons was performed by searching among a group of 32, 64, 128, 256, and 512 neurons and ReLU, ELU for the activation functions. Regarding the initial learning rate choice, we searched in an interval of 0.001 to 0.1 with the step of 0.002. Furthermore, the learning rate decay of 0.001 is used as the learning rate scheduler. All regularisation parameters, such as dropout rates, are chosen using the aforementioned framework.

Although the simulated time series have fixed and known lengths, that would not be generally expected in practice, and it is not true for the real dataset used in the evaluation. The difference in time series length is not a complication for the convolution layers, as they scan the time series locally. However, when we reach the classifier, the size of inputs must be fixed. To overcome this, we use a so-called global average pooling layer. At this point the output of the last pair of convolutional and pooling layers will be a matrix with 100 columns, each corresponding to a filter used for the convolution. The rows depend on the length of the time series, the number of pooling layers and their stride. Therefore, this is what we expect to vary when the number of observations in a time series changes. The global average pooling layer reduces each column to a single point by using the average, resulting in a vector with 100 elements, matching the number of filters used, irrespectively of the size of the time series. This information is passed to the classifier, thus resolving any sample size issues. This final level of abstraction results in the rich meta-features that the CNN meta learner can provide in a data driven fashion. We contrast this with conventional meta-learning, where the modeller selects the nature and number of meta-features, which are again collated in a vector prior to being passed to the classifier. The CNN produces automatically as many as the number of filters in the convolutional layers, in a data driven way that matches the time series used for training with the specific forecasts considered.

It should be noted that in order to be scalable, avoid overfitting, and have robust results, we used one structure in the design of CNNs in all of this study.

4.4.3 Benchmark Algorithms

We use four for model selection benchmarks, two based on information criteria and two on time series analysis using statistical testing: (i) AICc; (ii) BICc; (iii) statistical tests, Cox-Stuart and Friedman; and (iv) statistical tests Mann-Kendall and Friedman.

AICc and BICc are bias-corrected versions of AIC and BIC respectively, for small sample sizes (Burnham and Anderson, 2004). For larger samples, these become asymptotically the same as AIC and BIC, and therefore it is often recommended to use the corrected versions (Burnham and Anderson, 2004, Hyndman et al., 2008). Note that the ETS family of models with automatic model selection based on AIC has been found to perform very well on the M3 dataset (Hyndman et al., 2002, Hyndman et al., 2008).

The two selection algorithms based on statistical tests rely on the same seasonality test, but different trend tests. More specifically, for the trend tests, we use the Cox-Stuart (CS) and the Mann-Kendall (MK) tests (details in Appendix B). For the seasonality test, we rely on the Friedman (F) test, adapted to test whether the distribution of seasonal indices in a time series differs or not [see Appendix C, (Kourentzes, 2019b)]. Although there are many possible alternative tests, we rely on some of the most common ones, known for their robustness (Sun and Fang, 2017, Sayemuzzaman and Jha, 2014).

4.4.4 Evaluation scheme

We evaluate both the accuracy of selecting the appropriate model, but also the resulting forecast accuracy. As the candidate ETS models' parameters are optimised for each time series, similar models may perform well in terms of forecasting, even if only one is the correct generating model. Consider for example the case of ANN, the local level model, and AAN, the local trend model, where the latter can approximately behave like the former (Hyndman et al., 2008). This has been shown to be the case when selecting a model with information criteria, where different models may be picked by different criteria, which nonetheless perform similarly in terms of forecasting accuracy (Billah et al., 2006). Therefore, it is important to evaluate both aspects.

First, we evaluate the classification accuracy of the competing model selection approaches for the simulated time series. The classification accuracy provides the percentage of cases that a model is selected correctly by a selection methodology. We do not provide this metric for the real time series of the M3 competition, as the true data generating process is unknown.

To assess the forecast accuracy, we need to generate forecasts for each time series. To this end, we withhold the last 15 observations as a test set and produce rolling origin forecasts with a forecast horizon of 12-step ahead (Ord et al., 2017). This results in 3 forecasts per time series, exhausting the test set. At each origin the forecasting model is reoptimized. Moreover, these 15 observations are not used for model selection. All forecasts are generated using ETS models, as implemented in the smooth package (Svetunkov, 2017) for R.

We measure the forecast accuracy using the AvgRelMAE, proposed by Davydenko and Fildes (2013), due to its desirable statistical properties such as symmetric cost on positive and negative errors, as well as negligible chance to face computational issues. The AvgRelMAE is calculated as follows:

$$\text{AvgRelMAE} = \sqrt[n]{\prod_{j=1}^{n} \frac{\text{MAE}_j}{\text{MAE}_{j,b}}} \tag{4-4}$$

$$\text{MAE}_j\text{:}\ \frac{1}{h \times m} \sum_{i=1}^{m} \sum_{t=1}^{h} \left| y_{j,t} - \hat{y}_{j,t} \right| \tag{4-5}$$

where $y_{j,t}$ is the observation at period *t* for series *j*, $\hat{y}_{j,t}$ is the forecast at period *t* for series *j*, *n* is the number of time series that we summarise accuracy over, MAE$_j$ is the mean absolute error for time series *j* of forecast over *m* origins and *h*-step ahead and MAE$_{j,b}$ is the equivalent for a benchmark forecast.

We are interested in measuring the bias as well. For this, we rely on a modification of the AvgRelMAE, the AvgRelAME, that measures the absolute magnitude of bias in a prediction as:

$$\text{AvgRelAME} = \sqrt[n]{\prod_{j=1}^{n} \frac{\text{AME}_j}{\text{AME}_{j,b}}} \tag{4-6}$$

$$\text{AME}_j = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{h}\left|y_{j,t} - \hat{y}_{j,t}\right| \quad (4\text{-}7)$$

AME_j is the absolute mean error for time series j, of the forecast over m origins and h-steps ahead, and $\text{AME}_{j,b}$ is the equivalent for a benchmark forecast.

As a benchmark for both measures, we use the forecasting model selected by AICc for each time series. Both metrics are very simple to interpret. When their value is below 1 then the forecast is better than the benchmark and vice versa.

4.4.5 Model combination

For CNN, we calculate the weighted model combination with the weights being the probability of the forecasting models assigned by the network.

As benchmarks, we use the AICc and BICc combination weights. Given a set of P forecasting models, the combination weights (W_i) are calculated as:

$$\Delta AIC_i = AIC_i - min\, AIC(P) \quad i = 1,2,3,\dots,P$$

(4-8)

$$W_i = \frac{e^{(-0.5\,\Delta AIC_i)}}{\sum_{i=1}^{P} e^{(-0.5\,\Delta AIC_i)}} \quad (4\text{-}9)$$

A similar procedure is used for BICc combination weights.

4.4.6 Python package

For the purpose of providing a comprehensive framework of meta-learning in Python, we develop the MetaTS which is an open-source Python library designed to ease meta-learning for time series forecasting by offering a toolkit containing the typical components needed for a meta-learning workflow. In addition to provide new components and facilities, MetaTS aims to unify the available Python libraries which can be useful for meta-learning on time series data. For more information about the package, visit https://github.com/DrSasanBarak/metats.

4.5 Results of classification and forecasting with CNN and Benchmarks

4.5.1 Classification accuracy

Table 4.5-1 presents the classification accuracy for the 4- and 15-model cases, for the two different time series history length. Each column refers to a scenario, and the best performing selection approach highlighted in boldface.

In all cases, CNN performs best. First, we consider the 4-model case, where the AICc and BICc follow, and the statistical tests rank last. Note that the CNN and statistical tests display minimal differences for the different time series lengths. This is not the case for the information criteria which rely on having enough sample to estimate the model parameters adequately. Observe that the statistical tests perform rather poorly, even when the classification is between 4 quite distinct forecasts.

Shifting our attention to the 15-model case the overall classification accuracy drops substantially. This is to be expected given the similarity of the different models (see). Furthermore, even without the exact model choice, the forecasting performance may not be substantially harmed, as it depends on the optimized model parameters. For example, although the linear and the damped trend models are distinct, the resulting prediction can be fairly similar for some parameter ranges. The CNN remains first, with an accuracy of slightly over 55%, while the information criteria are closer to 50%.

Table 4.5-1. Classification accuracy of model selection

	4-model sample		**15-model sample**	
Classification accuracy %	48	144	48	144
AICc	90.65	93.82	43.60	50.10
BICc	93.47	94.57	52.26	52.44
CNN	**98.97**	**98.87**	**58.04**	**55.65**
CS-F	49.82	48.80	—	—
MK-F	64.65	60.92	—	—

In Table 4.5-1, we do not report the results of the MK-F and CS-F for the 15-model sample. The reason is the nature of MK-F and CS-F tests. Because they can only recognize trends and seasonality, they are divided into four classes depending on whether a time series has a trend/seasonality or not. Therefore, these algorithms can only be used to divide four classes and cannot separate 15 different classes.

Tables D-1 to D-4 in Appendix D, present the confusion matrices on the test set for the CNN for the 4 scenarios. The insight we gain from there is that ANN, which is a non-stationary model, is at times mixed with trend models. Furthermore, some issues appear between different trend types, i.e., linear and damped. As discussed in the design of the simulation, these issues are expected and are very challenging for all selection approaches considered.

4.5.2 Forecasting errors

Table 4.5-2 and Table 4.5-3 summarize the AvgRelMAE and the AvgRelAME for the 4 and 15 model selection scenarios, with the 44 and 144 time series length, for the simulated datasets. The tables provide the results for both the selection and the combination scenarios. For each column, the best selection and combination result is presented in boldface.

Table 4.5-4 shows that in all cases, the CNN chooses models that result in the lowest forecast errors, for selection or combination, across all scenarios. We observe that there are minimal differences between AICc and BICc, but also between the selection by statistical tests, where available. Given that the results are based on simulations, and the size can be increased arbitrarily, we do not test whether the differences are statistically significant. Table 4.5-5 presents the forecast bias magnitude results. In all but the 15-model and short history case the CNN ranks first. For that case, BICc performs particularly well. It is also interesting to observe that the reported similarities of the various benchmarks on forecast accuracy (Table 4.5-6) do not hold for the magnitude of the bias, in.

Table 4.5-2. AvgRelMAE for the 4-model and 15-model samples with 44 and 144 time series length

	4-model sample		15-model sample	
Model Choice	48	144	48	144
Selection				
AICc	1	1	1	1
BICc	0.983	1.015	1.026	0.991
CNN	**0.946**	**0.996**	**0.855**	**0.934**
CS-F	1.045	1.040	—	—
MK-F	1.027	1.029	—	—
Combination				
AICc Comb	1	1	1	1
BICc Comb	0.962	1.026	1.138	0.980
CNN Comb	**0.951**	**0.989**	**0.867**	**0.935**

Table 4.5-3. AvgRelAME for the 4-model and 15-model samples with 44 and 144 time series length

	4-model sample		15-model sample	
Model Choice	48	144	48	144
Selection				
AICc	1	1	1	1
BICc	0.932	0.986	**0.671**	0.854
CNN	**0.847**	**0.985**	0.959	**0.849**
CS-F	1.173	1.187	—	—
MK-F	1.145	1.185	—	—
Combination				
AICc Comb	1	1	1	1
BICc Comb	0.932	0.999	**0.708**	0.871
CNN Comb	**0.852**	**0.992**	0.964	**0.847**

Table 4.5-7 presents the overall results across the M3 monthly dataset. We can observe that although CNN is quite competitive, it does not rank first in

terms of AvgRelMAE, neither in the selection nor the combination tasks, with BICc being the overall best. Nonetheless, the CNN outperforms the AICc in the 4-model scenario. In terms of AvgRelAME, the CNN ranks first on the 4-model case and when selecting a model in the 15-model case.

Note that both AICc and BICc are choosing using the M3 data, while the CNN is trained in the simulated data and has not been trained in the richness of data structures observed in the M3 monthly data. In this respect, its forecasting performance is possible to improve with further additional training series. We argue that this is a worse case performance as all the simulated series, that the CNN was trained on, did not have any outliers or other effects present in the real time series. Nonetheless, it is possible to generate such cases and augment the performance of CNN, or alternatively use some of the real time series to fine tune the transfer learning for the CNN.

Table 4.5-4. Overall CNN forecasting results on M3 dataset

Model Choice	4 class		15 class	
	AvgRelMAE	AvgRelAME	AvgRelMAE	AvgRelAME
Selection				
AICc	1	1	**1**	1
BICc	**0.967**	0.767	1.002	**0.895**
CNN	0.978	**0.670**	1.051	0.942
CS-F	1.045	1.077	—	—
MK-F	1.142	1.125	—	—
Combination				
AICc Comb	1	1	**1**	1
BICc Comb	**0.974**	0.826	1.006	**0.900**
CNN combined	0.982	**0.707**	1.051	1.065

To further understanding the M3 results, we rely on the model selection achieved by AICc to characterize the real time series. Although the assigned labels do not correspond to the true data generation process, it provides some indication. To avoid overly influencing our analysis to the AICc selection, we further group together the labelled series into level, trend, seasonal, and trend-seasonal series, rather than the individual ETS models. Table 4.5-2 to Table 4.5-8 split the forecasting performance results for the M3 accordingly, for the 4-model and 15-model cases.

The tables are structured as follows. Each row corresponds to a time series type, labelled as described above. The columns further split the series into three groups: Short (≤ 48 observations), Medium (48 < observations ≤ 96), and Long (> 96 observations). For each case, we provide the performance of the AICc, BICc, and the proposed CNN. The best performing selection is highlighted in boldface.

In and that report the 4-model case, we observe that the CNN performs poorly for the series that exhibit some seasonal component. The result seems to improve as the time series length increases. It is also interesting to note that when comparing the overall performance over sample sizes, CNN is a strong contender for short series.

Table 4.5-5. AvgRelMAE for the 4-model sample by series type

Series Type	Short			Medium			Long		
	AICc	BICc	CNN	AICc	BICc	CNN	AICc	BICc	CNN
Level	1	1	**0.984**	1	1	1.031	1	1	1.042
Trend	1	0.899	**0.881**	1	0.889	0.942	1	0.957	**0.806**
Season	1	0.991	1.500	1	0.962	1.174	1	1.003	1.039
Trend-season	1	0.480	0.971	1	0.911	1.265	1	1.015	1.216
Overall	1	0.942	**0.949**	1	0.953	1.096	1	0.984	**0.955**

Table 4.5-6. AvgRelAME for the 4-model sample by series type

Series Type	Short			Medium			Long		
	AICc	BICc	CNN	AICc	BICc	CNN	AICc	BICc	CNN
Level	1	1	**0.979**	1	1	1.117	1	1	1.213
Trend	1	0.845	**0.783**	1	0.394	**0.305**	1	0.721	**0.281**
Season	1	0.668	0.824	1	1.123	1.358	1	0.816	0.929
Trend-season	1	0.650	0.913	1	0.392	0.374	1	0.905	1.032
Overall	1	0.906	**0.874**	1	0.598	**0.590**	1	0.792	**0.631**

Table 4.5-7 and Table 4.5-8 report the results for 15-model case, which largely mirror the ones reported for the 4-model case.

Table 4.5-7. AvgRelMAE for the 15-model sample by series type

Series Type	Short			Medium			Long		
	AICc	BICc	CNN	AICc	BICc	CNN	AICc	BICc	CNN
Level	1	1	**0.975**	1	1	**0.997**	1	1	**0.990**
Trend	1	0.917	0.958	1	0.982	1.004	1	1.011	**0.995**
Season	1	0.992	1.404	1	1.048	1.265	1	1.003	1.140
Trend-season	1	0.765	**0.698**	1	1.157	1.247	1	1.078	0.919
Overall	1	0.954	0.974	1	1.038	1.143	1	1.014	**0.991**

Table 4.5-8. AvgRelAME for the 15-model sample by series type

Series Type	Short			Medium			Long		
	AICc	BICc	CNN	AICc	BICc	CNN	AICc	BICc	CNN
Level	1	1	1.577	1	1	1.009	1	1	1.001
Trend	1	0.839	1.005	1	0.713	**0.634**	1	0.857	**0.828**
Season	1	0.825	1.736	1	1.048	1.282	1	0.847	0.907
Trend-season	1	0.847	**0.513**	1	0.595	0.875	1	1.131	**0.811**
Overall	1	0.902	1.248	1	0.809	0.878	1	0.891	**0.880**

4.6 Discussion

4.6.1 Stability of the CNN model selection

In this section, we look at the stability of model selection of CNN across forecast origins. CNN is not retrained and is simply re-run as new observation becomes available. We simulate four 480 length time series from models AAA, ANA, AAN, and ANA. We split each series in 433 sliding windows of 48 data points each and test the pre-trained CNN on each. We record the selection for all 433 windows and present the results in Table 4.6-1. This experiment explicitly evaluates the potential of the CNN to select a model consistently. Note that here we ensure adequate testing samples by simulating a very long time series, rather than many short ones. The benefit in this is that we can test the model selection as the patterns in the series evolve, given their stochastic generation process. The results show that for AAA and ANN, CNN can detect the correct model always, while for ANA and AAN, the classification accuracy is lower, determining a weakness of the network on seasonal series.

Table 4.6-1. Stability of the CNN model selection

Target	Correct model detection %
ANN	100
AAN	87.3
ANA	47.9
AAA	100

As mentioned, CNN cannot detect seasonality correctly, which is one of its limitations. This is because the CNN algorithm needs lots of training data to perform best. Since the size of the time series in the dataset is not large, this algorithm does not understand well whether the time series has seasonality and therefore has a problem detecting it. Of course, this limitation can be overcome by adding a dummy variable related to seasonality as an input to the model. In other words, we first detect seasonality using ETS models and give its output as a dummy variable to the CNN model. In this case, the accuracy of CNN in predicting seasonality will be improved.

4.6.2 Why the CNN works?

We have argued that the CNN imposes multiple transformations and aggregations of the input time series, abstracting rich meta-features.

Figure 4-2 illustrates the values of two consecutive convolutional layers, on the left and the right side of the picture, for an example series. The original series is plotted on the top-

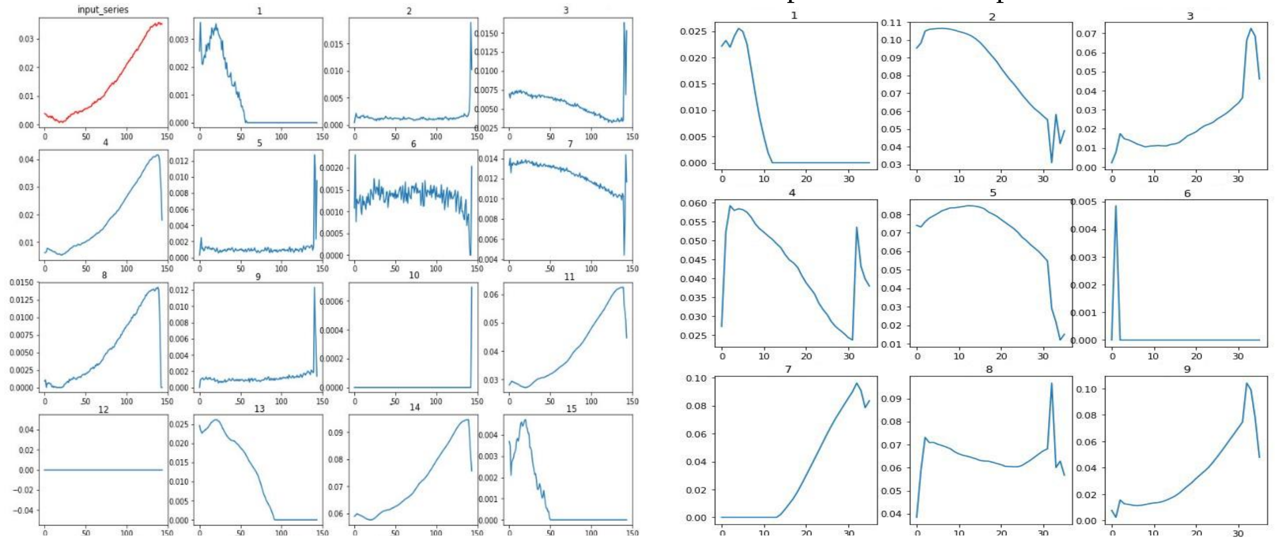


Figure 4-2. Feature representation of two convolutional layers

left panel. In the left side of the Figure 4-2, we provide 15 examples of filter outputs from the first convolutional layer, while the right side provides 9 examples of the second convolutional layer. Observe how the original time series is transformed into multiple different views, as well as how the second layer looks at more aggregate views. As the information is processed through the network, higher abstractions of the time Figure 4-2 series are generated, recording and summarizing the time series in different ways. We argue that although many may not be interpretable by the user, they carry useful information for the selection of the appropriate forecast.

More specifically we want to argue that why extracted and automated meta-features from CNN in compare with hand-crafted statistical meta-features are more suitable for model selection in terms of both quality and efficiency?

As an example, consider a set of time series randomly sampled from two different variants of the exponential smoothing family say ANN and MMM. We first extract a set of statistical features for our sample data using TsFresh (Christ et al., 2018) containing 426 features for each series. Figure 4-3 depicts the feature space after dimensionality reduction using principal component analysis (PCA). The x-axis and y-axis are PCA1 and PCA2 representing the direction of maximum variation through the data, respectively.

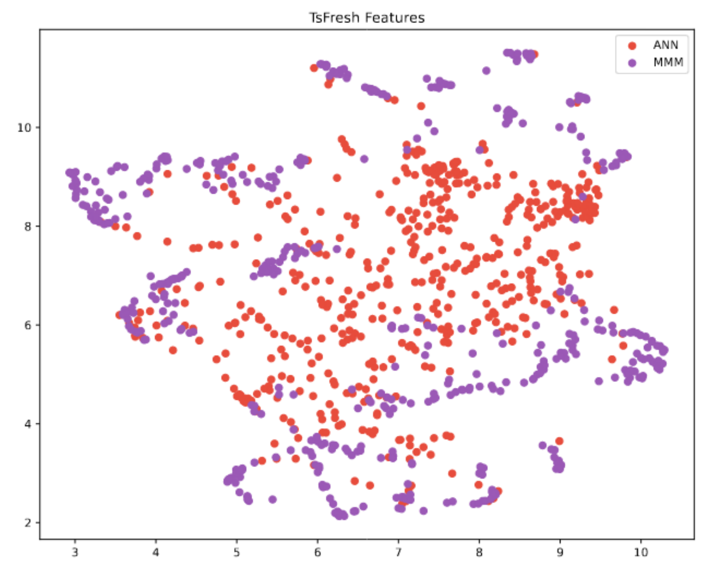
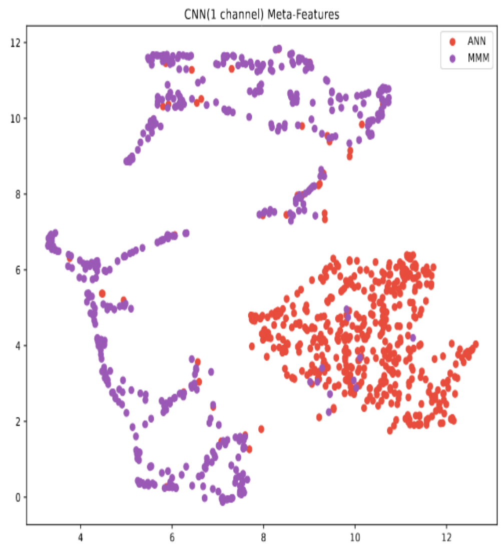


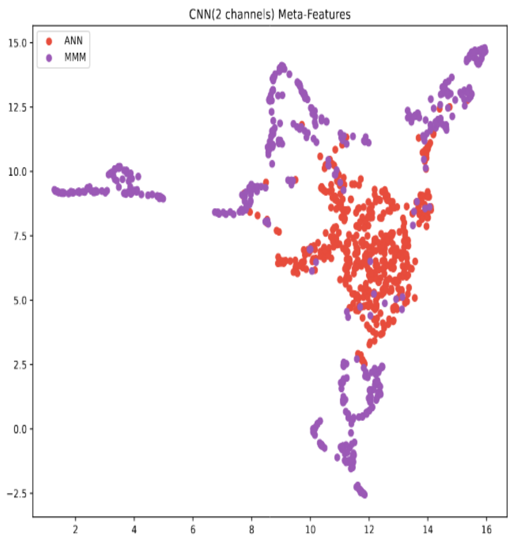
Figure 4-3. Variation of the exponential smoothing family

As you can see in the Figure 4-3, classifying the two models of the exponential smoothing family is not trivial in this space. The reason is that the statistical and hand-crafted features are not necessarily related to the data generation process.

On the other hand, since the objective of a CNN is to capture hidden patterns inside the series, the layers have to contain the information that is enough to explain the data. In other words, the needed information for mimicking the data generation process. Therefore, it is not surprising to see that the features extracted from CNN are more informative, especially for model selection. So, in the next step, we extract a set of meta-features for the same data using a

CNN with different number of channels. Figure 4-4 demonstrates the feature space of CNN with respectively 1, 2, 4, and 8 channels. As you can see, as the number of channels increases, two classes become more isolated in the feature space.





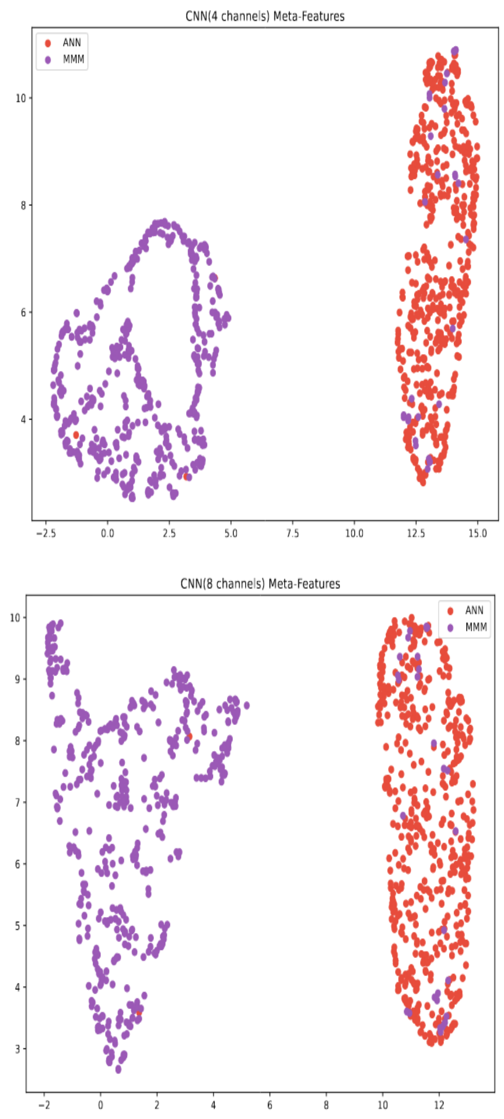


Figure 4-4. CNN meta space for different channels

Note in this example, the feature vector for each series extracted using TsFresh had a dimension of 426 while the dimension of extracted feature from CNN was 12 which is more efficient for any analysis including model selection.

4.7 Conclusion

In this chapter, we propose a novel use of deep learning in time series forecast selection. In contrast to the common model selection approaches, which are time-consuming in a big dataset or limited in the types of forecasts that can consider, the proposed CNN based model selection procedure is able to both select among a variety of forecasts that are not comparable via information criteria or similar, but also avoid resource constraints, due to the computational needs of cross-validation or similar approaches.

We train a CNN with one-dimensional convolutional layers on univariate time series for selecting amongst exponential smoothing models. The proposed CNN uses only past observations of the time series to assign an appropriate forecast to the time series. This is done by internally constructing features that help it match the available pool of forecasts to the provided time series data. We argue that this has a lot of similarities with meta-learning. However, our proposed approach has one important difference: it does not require the modeler to provide externally constructed meta-features. These are generated internally, not only simplifying and automating the process but also removing the need for the modeler to identify the appropriate set of meta-features. We postulate that this added flexibility allows it to match its internal representations as needed to best distinguish between the available forecasts.

Experimental results on both real and synthetic data show that the CNN performs very competitively against well-established approaches that use information criteria and statistical model selection. The former is often considered state-of-the-art for the exponential smoothing family of models. Due to the training needs for CNN, we rely on transfer learning to use it on real data. Although its performance is found to be promising, we identify a deficiency in detecting seasonal series. Further research should explore this further. Moreover, it would be pertinent to investigate further how to aid the learning of the network in the necessary transfer learning environment. Potential future studies include seeding the training set with real time series or introducing simulated examples of outliers and structural breaks.

5 Conclusion

This thesis analysed different meta-learning facets as an automated model selection approach for time series forecasting. Three meta-learning elements including the meta-features, the base learners, and the meta-learners (classifiers) were studied, and novel ideas to improve the overall forecast accuracy were presented.

In the second chapter, we investigated the idea of feature-based representation of time series and reviewed different groups of meta-features. Following this, we suggested a new group of meta-features based on statistical tests. To date, the use of statistical tests as meta-features has not been studied broadly in the literature; however, these tests have a solid background in time series analysis. Therefore, we explored their impact on the meta-learning accuracy and compared them with a prebuilt group of meta-features (tsfeatures). Moreover, the relationship between the meta-features, type of time series, and the base forecasters was evaluated to analyse the reason for obtaining good results in the meta-modelling. We empirically evaluated the new meta-features performance compared to commonly accepted model selection approaches such as information criteria and aggregated selection to test our approach's efficiency. We found that the selection process of base learners and meta-features needs intelligent design. The extracted meta-features in all meta-learning problems should be representative for the problem domain and forecast models. In the literature, none of the related studies justifies its reasons for choosing a different group of meta-features. This calls for more research into algorithmically choosing the most appropriate meta-features from a large pool of features.

In the third chapter, to explore the impact of different meta-learning elements, we studied meta-learning from three aspects: meta-feature, meta-learner, and the pool of base forecasters. Studying these elements helped us to understand their impact on the meta-learning performance individually or simultaneously. For the first perspective, we analysed different sets of meta-features from two prebuilt packages (Tsfresh, Tsfeatures) to study the features' importance in enhancing meta-learning accuracy. Second, to evaluate the meta-learners, we implemented three classification algorithms with single and ensemble properties. Finally, we applied two groups of base forecasters categorised as *simple* and *complex* forecasters. These models are categorised based on having internal model selection/combination features. Our finding shows that ensemble meta-learners, including random forest and XGBoost, perform best without excessive parameter tuning. Moreover, we argue that the Tsfresh, which has more extensive meta-features, outperforms the Tsfeatures, due to the type of features that are available in the Tsfresh. When we analysed all groups of features, we found that meta-features that used model characteristics (e.g., based on the statistical test, AR and MA of a ARIMA model) are more informative than the descriptive features (such as quantiles, mean, std, statistical summary). Since the Tsfeatures consist of many statistical description-based meta-features, its low performance is not unexpected.

In general, it was found that applying meta-learning as an automatic model selection outperformed all of the individual benchmark forecasters. Besides, we observed that simple base forecasters are more sensitive to the number of meta-feature groups. In contrast, dropping some of the least important feature groups increased the accuracy of meta-learning prediction for complex models.

Regarding base forecasters, we found that when all individual forecasters are chosen from a homogenous set of methods, meta-learning improves the overall model selection performance. However, meta-learning does not add value to the forecast accuracy when a base forecaster dominates all other forecasters. On the other hand, using complex and powerful base forecasters shows the potential of meta-learning in obtaining even more accuracy. Therefore, selecting a diverse and powerful pool of forecasters is essential to ensure the success of meta-learning.

Still, manually selected meta-features cannot satisfy our automated model selection aim. Due to high dependence on human judgement, and potentially low return of manually extracting features, it is helpful to find a method that can further reduce the human intervention and automate feature extracting and forecasting. In this case, deep CNN-based meta-learning proves to be a promising approach for capturing the time-series features and simultaneously forecasting them.

Therefore, in the fourth chapter, we proposed a novel deep meta-learning method based on Convolutional Neural Networks (CNNs). The advantage of the proposed approach is that it can automatically extract and learn features from time series. The common meta-learning approaches require manually defining and extracting features in an unsupervised way, which costs much time and requires expert knowledge in time series features. These requirements are unrealistic when faced with many time series from various business domains. Instead, CNN automates this complicated feature extracting process in meta-learning.

We trained the CNN using one-dimensional convolutional layers on univariate time series for selecting amongst exponential smoothing models. We employed the transfer learning idea to train the network on augmented synthetic data and evaluate it on real data. The time series augmentation was applied using a simulation package to enhance the sampling of ETS models. We increased the diversity of our simulated data generating a vast number of series using random model parameters.

Our findings showed that using our proposed deep meta-learning approach with transfer learning can achieve competitive results compared to the common meta-learning approaches. Our proposed deep meta-learning approach overcomes this challenge and manual feature generation and can be applied reliably in real-life business time series forecasting. Using augmented data to train our deep meta-learning approach dramatically increased the accuracy of our forecasting results. Having more representative simulated data could be a way to further improvement of the accuracy.

5.1 Implication for practice

Based on Alvarado-Valencia et al. (2017) and Petropoulos et al. (2018), time series methods are relatively intuitive, which makes them easy to apply by the end-users. On the contrary, machine learning and deep learning methods are often regarded as black boxes and are complex methods. They often provide limited or no insights into how the forecasts are formed and which features are essential. Because of this critical attribute, users and industry are mainly prone to use statistical and “glass” box methods. Although Meta-learning appears as an ML-based approach, it provides rule-based forecasting using the characteristics of the time series. In this way, the user can find the extracted rules from the meta-learner and understand the results by analysing the meta-features. Due to the high interpretability potential of meta learning-based model selection, users may find meta-learning easier to accept compared to standard (somewhat opaque) model selection approaches and forecast combinations. This user-study should be the focus of future work.

Meta learning can propose solutions for large-sized industry problems and mid-sized (or smaller) forecasting problems. We motivate the thesis within the realm of big data, but the majority of companies are not dealing with many time series. Moreover, one of the main challenges in training deep NN models is the lack of sufficient data. This thesis addresses this problem by providing a hybrid data augmentation and meta learning approach. The proposed transfer learning technique enhances the training rate of the time series and helps deep meta learners utilised in small-sized problems. An interesting question that remains is to what extend the success of transfer learning suggests that a company with a small to mid-sized forecasting problem can simply use a pre-trained meta-learner; this would overcome any data limitations.

In this thesis, we deliberately focus on exponential smoothing based models. These models are widely used in practice. In a survey of forecasting practices, this family used in more than one-third of times (32.1%) (Weller and Crone, 2012). Furthermore, Fildes et al., (2009) reveal an empirical study result that explains “the most common approach to forecasting demand in support of supply chain planning involves the use of a statistical software system which incorporates a simple univariate forecasting method, such as exponential smoothing, to produce an initial forecast”. It specifies that three-fourth of companies use variants of exponential smoothing methods. Therefore, applying meta learning in exponential smoothing variants can improve the forecasting practice in the industry. Due to the simplicity of meta-learning, its implementation in a software is easy, at the same time, open-source packages such as MetaTS allow users to make more advanced uses.

Finally, the growing willingness of industry to use open-source software makes deep NN applications more feasible in practice. This could help to move towards more automated model selection approaches such as meta-learning.

5.2 Limitations

One of the main problems in CNN implementation for time series forecasting is the variable length of the time series, which CNN cannot easily handle. There are some approaches such as zero paddings, global pooling; however, the efficacy of these approaches is data dependent. We overcome this problem using an augmentation approach, with our simulated time series being as long as the longest real time series. Thus, in the testing phase, we used zero padding at the beginning of the shorter time series to align with the training data length. This helps us address the time series length problem, but the zero numbers at the beginning of the time series may decrease the accuracy of the CNN forecasting. We tried to use a rolling window approach, but the differences were marginal. Analysing different approaches for handling time series of different length with deep neural networks remains unresolved. This is more important in CNN because of its high sensitivity to the size of the input data. We have not reached the best solution here, and it is still an open question for further research.

One of the unanswered questions here is the impact of the data on our results. Our methodology here is empirical research. This means that it is data-dependent and model-dependent. We used NN3 and M3, which are types of data on which exponential smoothing variants have performed better. Our augmented data is also based on exponential smoothing distribution. However, model selection from different data sources such as energy, call centres, finance, etc. is the ultimate goal of meta-learning. Although mentioned studies in the literature show that meta learning can outperform other approaches on a variety of data sources, using all those data sources together and training a meta-learner on all these different types of time series has not been investigated. Assessing the potential of the proposed model on different publicly available time series datasets such as the M4, M5, NN5, and NNGC and evaluating the effect of the number of time series in the meta-learning approach is one way to start this investigation. Moreover, our candidate forecasting algorithms are from very related families of models. Having less related models such as LSTMs or models with global training is of further interest.

Based on the amount of empirical evidence, we can argue that Meta-learning is a powerful approach in big data; however, there is not enough evidence that shows its potential for different types of data. Cross-domain effects of time series on each other have not been analysed in this study. Maybe the cross-domain transfer of meta-knowledge is a key for successfully forecasting multiple data sources in meta-learning.

We used two sets of pre-made features. In the third chapter, we went beyond these two groups and evaluated the impact of different meta-features on forecasting accuracy. We empirically demonstrated the successful types of features by analysing the characteristics of the features. The results may hold for different sets of features, but it is not a given. In this thesis we used to extensive sets of features, but these do not cover all possible types of specifications of features. Nonetheless, we argue that the approach we followed in the analysis here can be easily extended to include more features and investigate the usefulness of different classes of features not used here.

Having an ensemble meta-learner instead of an individual classifier may improve classification accuracy. There is a direct relationship between the forecast accuracy and the classifier (meta-learner) accuracy in many meta-learning implementations. Therefore, a fusion of models as an ensemble meta-learner can increase the classifier competence. We did not use ensemble meta learner; however, it is worth noting that having more sophisticated classifiers in small-sized problems may lead to overfitting. Again, data augmentation and transfer learning may be helpful approaches to overcome data limitations.

Another important aspect of this thesis is related to the extracted features from the CNN. We interpret these as meta-features, but we do not show how these correlates, if at all, with existing features, and we do not show what style of meta-features the CNN extracted. This is a limitation because the features extracted by CNN although perform well, are like a black box that lessen our interpretation power. The meaning of the power of interpretability is that, unlike the features that are extracted through statistical tests, we cannot talk about the nature of these features.

Although we have used several methods, such as dropout and batch normalisation, which have been widely accepted in preventing overfitting for CNNs, the overfitting issue still partially exists in our proposed deep meta-learning approach. This may also connect to the relatively worse performance of the CNN on seasonal time series.

5.3 Further Research

In our discussion so far, we already identified a number of open questions for future research. Here we list a number of potentially fruitful avenues of research in addition to the elements identified before.

Exploring the potential of applying other state-of-the-art architectures of CNNs in our proposed deep meta-learning approach, such as Inception-v4 (Szegedy et al., 2016) and Resnet (He et al., 2016). These are developed versions of the CNN model, and they show their superiority in image recognition tasks. However, there is a limited study to implement these models in time series forecasting and their potential in a one-dimensional convolution are not clearly analysed. We argue that some of the features of these innovations can potentially help generate richer meta-features. Similarly, implementing data augmentation using deep approaches such as Generative Adversarial Network and Variational Autoencoders. These approaches generate the time series using the extracted meta-features in the latent space of the time series and may better support a deep meta-learning approach.

Remaining on the modelling front, the main focus of this work has been on the model selection, matching the common approach to forecasting in practice. Selecting a single time series model is often helpful for the users, especially when this is a well understood forecasting model, as it increases the trust in the forecasts. Nonetheless, there is a overwhelming evidence that forecast combination can result in more stable and accurate forecasts. Evaluating the performance of our proposed deep meta-learning approach for forecasting model combination using other loss functions, such as weighted average loss function could be useful. In general, more focus on model averaging than model selection could be useful.

In terms of the base forecaster models, applying less related models such as LSTM and RNN to the current forecasting algorithms and evaluating their impact on the accuracy could be useful. We identified that when there is a substantially better base forecaster then meta-learning adds limited value. In this work we used models fitted to individual time series only. It is of interest to understand how this finding holds when we use global forecasting models, trained on multiple time series, whether this is conditional on individually trained models, as the global models are typically not optimal on any single time series, in the usual minimum fitting error sense.

Finally, in terms of data we already identified the desirability for using a variety of time series domains, especially for instance for the seasonal case. We also note that training with simulated data needs further investigation for the parameter tuning of the generator. Here we simulated data that reasonably matched the real times series. However, this may in fact limit the learning of the CNN and a more diverse set of parameters for simulating time series could be beneficial. This, for example, may also resolve the poor performance in seasonal data. Furthermore, adding simulated data of outliers and structural breaks and evaluating the potential of meta-learning in this data is of interest, particularly when these are very common in some forecasting applications, such as retailing.

Overall, meta-learning is a very promising alternative to conventional model selection methodologies. We have identified a number of ways to progress the research based on the findings of this thesis. We argue that although standard meta-learning works well, leveraging the potential of deep learning and CNNs can lead to exciting future research.

6 Appendices

Appendix A. Features classification based on their calculation method

Table A-1. Features classification based on their calculation method

Feature group	Features
Dickey Fuller	2 features including augmented dickey fuller “pvalue” and “teststat” attributes. 1. value__augmented_dickey_fuller__attr_"pvalue" 2. value__augmented_dickey_fuller__attr_"teststat"
Auto Regressive	2 features extracted from autoregressive process, including AR(10) first and second coefficients. 1. value__ar_coefficient__k_10__coeff_0 2. value__ar_coefficient__k_10__coeff_1
Auto Correlation	19 features obtained from: • the differenced series of autocorrelation function (lag 1 and 10); • the twice-differenced series of autocorrelation function (lag 1 and 10); • error autocorrelation function for lags 1 and 10 • seas_acf1: the sum of squares of the first 10 autocorrelation coefficients • the variance and median of aggregate autocorrelation vectors of lags 1 to 9; • autocorrelation value for lags 2 and 4 • partial autocorrelation of the time series (lags of 1 to 9); 1. diff1_acf1 2. diff1_acf10 3. diff2_acf1 4. diff2_acf10 5. e_acf1 6. e_acf10 7. seas_acf1 8. value__agg_autocorrelation__f_agg_"median" 9. value__agg_autocorrelation__f_agg_"var" 10. value__autocorrelation__lag_2 11. value__autocorrelation__lag_4 12. value__partial_autocorrelation__lag_1 13. value__partial_autocorrelation__lag_2 14. value__partial_autocorrelation__lag_3 15. value__partial_autocorrelation__lag_4 16. value__partial_autocorrelation__lag_5 17. value__partial_autocorrelation__lag_7 18. value__partial_autocorrelation__lag_8

	19. value__partial_autocorrelation__lag_9
Energy	10 features calculated as the sum of squares of chunk I (for I from 1 to 10) out of 10 chunks expressed as a ratio with the sum of squares over the whole series. 1. value__abs_energy 2. value__energy_ratio_by_chunks__num_segments_10__segment_focus_0 3. value__energy_ratio_by_chunks__num_segments_10__segment_focus_1 4. value__energy_ratio_by_chunks__num_segments_10__segment_focus_3 5. value__energy_ratio_by_chunks__num_segments_10__segment_focus_4 6. value__energy_ratio_by_chunks__num_segments_10__segment_focus_5 7. value__energy_ratio_by_chunks__num_segments_10__segment_focus_6 8. value__energy_ratio_by_chunks__num_segments_10__segment_focus_7 9. value__energy_ratio_by_chunks__num_segments_10__segment_focus_8 10. value__energy_ratio_by_chunks__num_segments_10__segment_focus_9
Entropy	5 features including Shanon entropy, binned entropy calculated based on different equidistant bins of time series, and cross power spectral density of the time series at different frequencies (3 coefficients). 1. Shanon entropy 2. value__binned_entropy__max_bins_10 3. value__spkt_welch_density__coeff_2 4. value__spkt_welch_density__coeff_5 5. value__spkt_welch_density__coeff_8
Wavelet Transformation	12 features of continuous wavelet transformation (cwt) considering multiple values for two different parameters including : widths, coefficients. 1. value__cwt_coefficients__widths_(2,5,10,20)__coeff_0__w_2 2. value__cwt_coefficients__widths_(2,5,10, 0)__coeff_0__w_5 3. value__cwt_coefficients__widths_(2,5,10, 20)__coeff_13__w_5 4. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_2 5. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_20 6. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_2__w_2 7. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_2__w_5 8. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_4__w_2 9. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_5__w_2 10. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_6__w_2 11. value__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_7__w_10 12. value__number_cwt_peaks__n_1
Fourier Transformation	35 features are extracted from the one-dimensional discrete Fourier Transform by fast Fourier transformation algorithm including real part (attr="real"), the imaginary part (attr="imag"), the absolute value (attr="abs") and the angle in degrees (attr="angle") (35 features); and 4 features including spectral centroid (mean), variance, skew, and kurtosis of the absolute Fourier transform spectrum. 1. value__fft_coefficient__coeff_0__attr_"abs" 2. value__fft_coefficient__coeff_1__attr_"abs" 3. value__fft_coefficient__coeff_1__attr_"angle" 4. value__fft_coefficient__coeff_1__attr_"imag" 5. value__fft_coefficient__coeff_1__attr_"real"

	6. value__fft_coefficient__coeff_10__attr_"abs"
	7. value__fft_coefficient__coeff_10__attr_"angle"
	8. value__fft_coefficient__coeff_10__attr_"real"
	9. value__fft_coefficient__coeff_2__attr_"abs"
	10. value__fft_coefficient__coeff_2__attr_"angle"
	11. value__fft_coefficient__coeff_2__attr_"imag"
	12. value__fft_coefficient__coeff_2__attr_"real"
	13. value__fft_coefficient__coeff_3__attr_"abs"
	14. value__fft_coefficient__coeff_3__attr_"angle"
	15. value__fft_coefficient__coeff_3__attr_"imag"
	16. value__fft_coefficient__coeff_4__attr_"abs"
	17. value__fft_coefficient__coeff_4__attr_"angle"
	18. value__fft_coefficient__coeff_4__attr_"imag"
	19. value__fft_coefficient__coeff_5__attr_"abs"
	20. value__fft_coefficient__coeff_5__attr_"angle"
	21. value__fft_coefficient__coeff_5__attr_"imag"
	22. value__fft_coefficient__coeff_5__attr_"real"
	23. value__fft_coefficient__coeff_6__attr_"abs"
	24. value__fft_coefficient__coeff_6__attr_"angle"
	25. value__fft_coefficient__coeff_6__attr_"imag"
	26. value__fft_coefficient__coeff_7__attr_"abs"
	27. value__fft_coefficient__coeff_7__attr_"angle"
	28. value__fft_coefficient__coeff_7__attr_"imag"
	29. value__fft_coefficient__coeff_8__attr_"abs"
	30. value__fft_coefficient__coeff_8__attr_"angle"
	31. value__fft_coefficient__coeff_8__attr_"real"
	32. value__fft_coefficient__coeff_9__attr_"abs"
	33. value__fft_coefficient__coeff_9__attr_"angle"
	34. value__fft_coefficient__coeff_9__attr_"imag"
	35. value__fft_coefficient__coeff_9__attr_"real"
	36. value__fft_aggregated__aggtype_"centroid"
	37. value__fft_aggregated__aggtype_"variance"
	38. value__fft_aggregated__aggtype_"kurtosis"
	39. value__fft_aggregated__aggtype_"skew"
STL	19 features including linearity, trend, seasonal_period, and seasonal_strength calculated based on the coefficients of an orthogonal quadratic regression; Also STL decomposed features extracted from linear least-squares regression for values of the time series that were aggregated over (10 and 5) chunks, consisting of “pvalue”, “R square value (rvalue)”, “intercept”, “slope”, “standard error (stderr)” of linear least-squares regression for values of the time series that were aggregated over (10 and 5) chunks.
	1. linearity
	2. seasonal_period
	3. seasonal_strength

	4. trend 5. value__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"stderr" 6. value__agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"stderr" 7. value__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"intercept" 8. value__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"slope" 9. value__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"stderr" 10. value__agg_linear_trend__f_agg_"min"__chunk_len_5__attr_"rvalue" 11. value__agg_linear_trend__f_agg_"min"__chunk_len_5__attr_"stderr" 12. value__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"rvalue" 13. value__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"stderr" 14. value__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"rvalue" 15. value__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"slope" 16. value__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"stderr" 17. value__c3__lag_3 18. value__linear_trend__attr_"pvalue" 19. value__linear_trend__attr_"stderr"
Quantiles	23 features related to a corridor selected by specific values for low and high quantiles of the time series; Features are calculated as the outputs of applying 1. value__change_quantiles__f_agg_"mean"__isabs_True__qh_0.2__ql_0.0 2. value__change_quantiles__f_agg_"mean"__isabs_True__qh_0.4__ql_0.2 3. value__change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.4 4. value__change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.4 5. value__change_quantiles__f_agg_"mean"__isabs_True__qh_0.8__ql_0.6 6. value__change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.8 7. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.2__ql_0.0 8. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.4__ql_0.2 9. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.0 10. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.2 11. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.6__ql_0.4 12. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.2 13. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.4 14. value__change_quantiles__f_agg_"var"__isabs_False__qh_0.8__ql_0.6 15. value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.0 16. value__change_quantiles__f_agg_"var"__isabs_False__qh_1.0__ql_0.2 17. value__change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.0 18. value__change_quantiles__f_agg_"var"__isabs_True__qh_0.4__ql_0.2 19. value__change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.4 20. value__change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.4 21. value__change_quantiles__f_agg_"var"__isabs_True__qh_0.8__ql_0.6 22. value__index_mass_quantile__q_0.1 23. value__index_mass_quantile__q_0.9 different aggregator function (mean and variance) on consecutive changes of values in each corridor. The change values can be considered as absolute or not.
Descriptive Statistics	38 features including statistical summary of the time series such as mean, variance, minimum, maximum in conjunction with other feature related to distribution of data

1. curvature
2. nperiods
3. peak
4. spike
5. trough
6. value__ complexity-invariant distance for time series
7. value__count_above_mean
8. value__has_duplicate
9. value__has_duplicate_min
10. value__kurtosis
11. value__large_standard_deviation__r_0.15
12. value__large_standard_deviation__r_0.2
13. value__large_standard_deviation__r_0.25
14. value__large_standard_deviation__r_0.30
15. value__large_standard_deviation__r_0.35
16. value__last_location_of_minimum
17. value__longest_strike_above_mean
18. value__longest_strike_below_mean
19. value__maximum
20. value__minimum
21. value__mean_second_derivative_central
22. value__mean_abs_change
23. value__number_peaks__n_10
24. value__number_peaks__n_5
25. value__percentage_of_reoccurring_datapoints_to_all_datapoints
26. value__ratio_beyond_r_sigma__r_0.5
27. value__ratio_beyond_r_sigma__r_1
28. value__ratio_beyond_r_sigma__r_2
29. value__ratio_beyond_r_sigma__r_2.5
30. value__ratio_beyond_r_sigma__r_3
31. value__ratio_beyond_r_sigma__r_5
32. value__skewness
33. value__standard_deviation
34. value__sum_of_reoccurring_data_points
35. value__symmetry_looking__r_0.05
36. value__symmetry_looking__r_0.1
37. value__time_reversal_asymmetry_statistic__lag_3
38. value__variance

Appendix B. Trend tests

B1. Cox-Stuart test

The Cox-Stuart test non-parametrically defines whether there is a change in the mean of the series. Given a set of ordered observations $X_1, X_2, ..., X_n$, let

$$c = \frac{n}{2} \quad if\ n \quad even \tag{B-1}$$

$$= \frac{(n+1)}{2} \quad if\ n\ odd$$

We split the time series into two parts prior and after c. Then pair the data as $X_1, X_{1+c}, X_2, X_{2+c}, ..., X_{n-c}, X_n$. The Cox-Stuart test applies a sign test to the paired data (Hollander et al., 2013). Cox-Stuart test is applied to the de-seasonalized time series with tsutils package (Kourentzes, 2019a) in R.

B2. Mann-Kendall test

Mann-Kendall trend test with the null hypothesis of "no trend" (MK) is calculated according to:

$$S = \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} \operatorname{sgn}(X_j - X_k) \tag{B-2}$$

where $$sgn(x) = \begin{cases} 1 & if \quad X > 0 \\ 0 & if \quad X = 0 \\ -1 & if \quad X < 0 \end{cases} . \tag{B-3}$$

The statistic S is closely related to Kendall's τ, as given by $\tau = S / D$ where

$$D = \left[\frac{1}{2}n(n-1) - \frac{1}{2}\sum_{j=1}^{p} t_j (t_j - 1)\right]^{1/2} \left[\frac{1}{2}n(n-1)\right]^{1/2} \tag{B-4}$$

P is the number of the tied groups in the data set, and t_j is the number of data points in the j^{th} tied group (Pohlert, 2016). Mann-Kendall trend test is applied to the deseasonalized time series with Trend package (Pohlert et al., 2016) in R.

Appendix C. Seasonality test

C1. Friedman test

The test is a nonparametric alternative to ANOVA, and we apply it to the de-trended time series' seasonal indices, as extracted by classical decomposition (Hyndman and Athanasopoulos, 2018). No normality assumption is required for the distributions of the seasonal indices. The Friedman statistic Q is given by

$$Q = \frac{12}{nK(K+1)} SS_{col}^{'} \ , \tag{C-1}$$

where SS'_{col} is the sum of squares between groups using the ranks instead of raw data. The Friedman test assumes that there are K experimental treatments ($K \geq 2$), and the observations are arranged in n blocks. When $K \geq 5$, the probability distribution of Q can be approximated by that of a chi-squared distribution and the null hypothesis is rejected when $Q > \chi^2_{Critical}$. The test is applied with tsutils package (Kourentzes, 2019a) in R.

Appendix D. CNN confusion matrix

Table D-1. CNN confusion matrix for 144 length time series with 15 classes

T\P	ANN	MNN	AAN	AADN	MAN	MADN	ANA	MNA	AAA	AADA	MAA	MADA	MNM	MAM	MADM
ANN	**195**	0	42	4	1	0	2	0	0	8	0	3	10	0	3
MNN	1	**46**	0	203	0	2	0	0	0	0	0	0	0	0	1
AAN	178	1	**58**	1	1	0	6	1	0	10	0	1	4	1	4
AADN	0	33	0	**210**	0	1	0	0	0	0	0	0	0	0	0
MAN	0	0	0	0	**178**	0	0	0	0	0	0	0	24	23	2
MADN	0	2	0	0	0	**207**	0	0	0	0	0	0	0	0	35
ANA	12	0	8	0	2	0	**97**	2	5	69	1	26	10	0	0
MNA	0	0	0	0	0	0	1	**191**	1	0	45	0	0	1	0
AAA	1	0	1	1	0	0	0	0	**111**	7	0	139	0	0	3
AADA	20	0	20	0	2	0	53	0	12	**116**	0	39	12	0	0
MAA	0	0	0	0	0	0	1	217	0	0	**60**	0	0	2	0
MADA	0	1	0	0	0	1	3	0	90	7	0	**110**	0	0	9
MNM	3	0	1	0	155	0	0	0	0	0	0	0	**71**	17	0
MAM	0	0	0	0	31	6	0	11	0	0	0	0	3	**189**	2
MADM	0	0	0	0	0	1	0	0	0	0	0	1	0	0	**248**

Accuracy on test data: 55.65%

Table D-2. CNN confusion matrix for 48 length time series with 15 classes

T/P	ANN	MNN	AAN	AAdN	MAN	MAdN	ANA	MNA	AAA	AAdA	MAA	MAdA	MNM	MAM	MAdM
ANN	**469**	27	0	0	0	0	0	0	0	0	0	0	3	1	0
MNN	24	**468**	0	2	5	1	0	0	0	0	0	0	0	0	0
AAN	1	5	**247**	238	7	2	0	0	0	0	0	0	0	0	0
AAdN	1	6	236	**239**	15	2	0	0	0	0	0	0	0	1	0
MAN	0	8	0	3	**431**	39	0	0	0	2	15	2	0	0	0
MAdN	2	9	1	4	396	**56**	0	0	1	0	27	4	0	0	0
ANA	0	0	0	0	0	0	**429**	62	0	1	0	0	8	0	0
MNA	0	0	0	0	0	0	136	**278**	2	2	2	9	71	0	0
AAA	0	0	1	0	1	0	2	6	**231**	244	2	12	1	0	0
AAdA	0	0	0	0	2	0	2	14	198	**256**	7	21	0	0	0
MAA	0	0	0	0	22	2	9	60	16	10	**226**	151	10	0	4
MAdA	0	0	0	0	21	5	10	59	8	4	165	**211**	8	3	6
MNM	4	3	0	0	0	0	17	42	0	0	0	0	**416**	1	17
MAM	2	0	0	1	0	0	0	4	0	0	4	1	67	**47**	374
MAdM	1	0	0	0	0	0	0	5	0	0	4	3	79	59	**349**

classification accuracy: 58.04 %

Table D-3. CNN Confusion Matrix for 48 length time series with 4 classes

T\P	**ANN**	**ANA**	**AAA**	**AAN**
ANN	998	0	0	2
ANA	0	1000	0	0
AAA	0	25	973	2
AAN	12	0	0	**988**

Table D-4. CNN Confusion Matrix for 144 length time series with 4 classes

T\P	**AAA**	**AAN**	**ANA**	**ANN**
AAA	**972**	5	24	0
AAN	6	**982**	0	2
ANA	8	0	**1010**	0
ANN	0	0	0	991

7 References

ADYA, M., COLLOPY, F., ARMSTRONG, J. S. & KENNEDY, M. 2001. Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting,* 17**,** 143-157.

ALVARADO-VALENCIA, J., BARRERO, L. H., ÖNKAL, D. & DENNERLEIN, J. T. 2017. Expertise, credibility of system forecasts and integration methods in judgmental demand forecasting. *International Journal of Forecasting,* 33**,** 298-313.

ARINZE, B. 1994. Selecting appropriate forecasting models using rule induction. *Omega,* 22**,** 647-658.

ARMSTRONG, J. S. 2006. Findings from evidence-based forecasting: Methods for reducing forecast error. *International Journal of Forecasting,* 22**,** 583-598.

ASSIMAKOPOULOS, V. & NIKOLOPOULOS, K. 2000. The theta model: a decomposition approach to forecasting. *International journal of forecasting,* 16**,** 521-530.

ATHANASOPOULOS, G., HYNDMAN, R. J., KOURENTZES, N. & PETROPOULOS, F. 2017. Forecasting with temporal hierarchies. *European Journal of Operational Research,* 262**,** 60-74.

BAGNALL, A., DAVIS, L., HILLS, J. & LINES, J. Transformation based ensembles for time series classification. Proceedings of the 2012 SIAM international conference on data mining, 2012. SIAM, 307-318.

BAGNALL, A., LINES, J., BOSTROM, A., LARGE, J. & KEOGH, E. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery,* 31**,** 606-660.

BARAK, S., ARJMAND, A. & ORTOBELLI, S. 2017. Fusion of multiple diverse predictors in stock market. *Information Fusion,* 36**,** 90-102.

BARAK, S., DAHOOIE, J. H. & TICHÝ, T. 2015. Wrapper ANFIS-ICA method to do stock market timing and feature selection on the basis of Japanese Candlestick. *Expert Systems with Applications,* 42**,** 9221-9235.

BENGIO, Y., COURVILLE, A. & VINCENT, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence,* 35**,** 1798-1828.

BERGMEIR, C., HYNDMAN, R. J. & BENÍTEZ, J. M. 2016. Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation. *International journal of forecasting,* 32**,** 303-312.

BILLAH, B., KING, M. L., SNYDER, R. D. & KOEHLER, A. B. 2006. Exponential smoothing model selection for forecasting. *International journal of forecasting,* 22**,** 239-247.

BLUM, A. & HARDT, M. The ladder: A reliable leaderboard for machine learning competitions. International Conference on Machine Learning, 2015. 1006-1014.

BOX, G. E., JENKINS, G. M., REINSEL, G. C. & LJUNG, G. M. 2015. *Time series analysis: forecasting and control,* John Wiley & Sons.

BRAZDIL, P., CARRIER, C. G., SOARES, C. & VILALTA, R. 2008. *Metalearning: Applications to data mining,* Springer Science & Business Media.

BREIMAN, L. 1996. Bagging predictors. *Machine learning,* 24**,** 123-140.

BREIMAN, L. 2001. Random forests. *Machine learning,* 45**,** 5-32.

BROCKWELL, P. J. & DAVIS, R. A. 2016. *Introduction to time series and forecasting*, springer.

BURNHAM, K. P. & ANDERSON, D. R. 2002. A practical information-theoretic approach. *Model selection and multimodel inference, 2nd ed. Springer, New York*.

BURNHAM, K. P. & ANDERSON, D. R. 2003. *Model selection and multimodel inference: a practical information-theoretic approach*, Springer Science & Business Media.

BURNHAM, K. P. & ANDERSON, D. R. 2004. Multimodel inference: understanding AIC and BIC in model selection. *Sociological methods & research,* 33**,** 261-304.

CERQUEIRA, V., TORGO, L., PINTO, F. & SOARES, C. Arbitrated ensemble for time series forecasting. Joint European conference on machine learning and knowledge discovery in databases, 2017. Springer, 478-494.

CHAN, F. & PAUWELS, L. L. 2018. Some theoretical results on forecast combinations. *International Journal of Forecasting,* 34**,** 64-74.

CHEN, T. & GUESTRIN, C. Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016. ACM, 785-794.

CHEN, T., HE, T. & BENESTY, M. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2***,** 1-4.

CHOLLET, F. 2015. keras.

CHRIST, M., BRAUN, N., NEUFFER, J. & KEMPA-LIEHR, A. W. 2018. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing,* 307**,** 72-77.

CHU, C.-H. & WIDJAJA, D. 1994. Neural network system for forecasting method selection. *Decision Support Systems,* 12**,** 13-24.

CLEVELAND, R. B., CLEVELAND, W. S., MCRAE, J. E. & TERPENNING, I. 1990. STL: A seasonal-trend decomposition. *Journal of official statistics,* 6**,** 3-73.

COLLOBERT, R. & WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. Proceedings of the 25th international conference on Machine learning, 2008. ACM, 160-167.

CRONE, S. F., HIBON, M. & NIKOLOPOULOS, K. 2011. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of forecasting,* 27**,** 635-660.

CUI, C., HU, M., WEIR, J. D. & WU, T. 2016a. A recommendation system for meta-modeling: A meta-learning based approach. *Expert Systems with Applications,* 46**,** 33-44.

CUI, Z., CHEN, W. & CHEN, Y. 2016b. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*.

DAVYDENKO, A. & FILDES, R. 2013. Measuring forecasting accuracy: The case of judgmental adjustments to SKU-level demand forecasts. *International Journal of Forecasting,* 29**,** 510-522.

DE LIVERA, A. M., HYNDMAN, R. J. & SNYDER, R. D. 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association,* 106**,** 1513-1527.

DIMITRIADOU, E., HORNIK, K., LEISCH, F., MEYER, D., WEINGESSEL, A. & LEISCH, M. F. 2006. The e1071 package. *Misc Functions of Department of Statistics (e1071), TU Wien*.

DOS SANTOS, P. M., LUDERMIR, T. B. & PRUDENCIO, R. B. C. Selection of time series forecasting models based on performance information. Hybrid Intelligent Systems, 2004. HIS'04. Fourth International Conference on, 2004. IEEE, 366-371.

ELLIOTT, G. & TIMMERMANN, A. 2016. *Economic Forecasting,* Princeton University Press

FAN, J., WANG, X., WU, L., ZHOU, H., ZHANG, F., YU, X., LU, X. & XIANG, Y. 2018. Comparison of Support Vector Machine and Extreme Gradient Boosting for predicting daily global solar radiation using temperature and precipitation in humid subtropical climates: A case study in China. *Energy Conversion and Management,* 164**,** 102-111.

FAWCETT, T. 2006. An introduction to ROC analysis. *Pattern recognition letters,* 27**,** 861-874.

FILDES, R. 1989. Evaluation of aggregate and individual forecast method selection rules. *Management Science,* 35**,** 1056-1065.

FILDES, R. 1992. The evaluation of extrapolative forecasting methods. *International Journal of Forecasting,* 8**,** 81-98.

FILDES, R. 2001. Beyond forecasting competitions. *International Journal of Forecasting,* 17**,** 556-560.

FILDES, R., MA, S. & KOLASSA, S. 2019. Retail forecasting: Research and practice. *International Journal of Forecasting*.

FILDES, R., MADDEN, G. & TAN, J. 2007. Optimal forecasting model selection and data characteristics. *Applied Financial Economics,* 17**,** 1251-1264.

FILDES, R. & PETROPOULOS, F. 2015. Simple versus complex selection rules for forecasting many time series. *Journal of Business Research,* 68**,** 1692-1701.

FIORUCCI, J. A., LOUZADA, F., YIQI, B. & FIORUCCI, M. J. A. 2016. Package 'forecTheta'.

FRIEDMAN, J., HASTIE, T. & TIBSHIRANI, R. 2001. *The elements of statistical learning,* Springer series in statistics New York.

FULCHER, B. D. 2017. Feature-based time-series analysis. *arXiv preprint arXiv:1709.08055*.

FULCHER, B. D. & JONES, N. S. 2014. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering,* 26**,** 3026-3037.

FULCHER, B. D., LITTLE, M. A. & JONES, N. S. 2013. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of The Royal Society Interface,* 10**,** 20130048.

GARDNER JR, E. S. 1985. Exponential smoothing: The state of the art. *Journal of forecasting,* 4**,** 1-28.

GARDNER JR, E. S. 2006. Exponential smoothing: The state of the art—Part II. *International journal of forecasting,* 22**,** 637-666.

GARDNER JR, E. S. & MCKENZIE, E. 1988. Model identification in exponential smoothing. *Journal of the Operational Research Society***,** 863-867.

GIRAUD-CARRIER, C. Metalearning-a tutorial. Proceedings of the 7th international conference on machine learning and applications, 2008a.

GIRAUD-CARRIER, C. Metalearning-a tutorial. Tutorial at the 2008 International Conference on Machine Learning and Applications, ICMLA, 2008b. 11-13.

GOMES, J. & VELHO, L. 2015. The Fast Wavelet Transform. *From Fourier Analysis to Wavelets.* Springer.

GOODFELLOW, I., BENGIO, Y. & COURVILLE, A. 2016. *Deep learning,* MIT press.

GOODRICH, R. L. 1992. *Applied statistical forecasting,* Business Forecast Systems, Incorporated.

HAMED, K. H. 2008. Trend detection in hydrologic data: the Mann–Kendall trend test under the scaling hypothesis. *Journal of hydrology,* 349**,** 350-363.

HARVEY, D. Y. & TODD, M. D. 2015. Automated feature design for numeric sequence classification by genetic programming. *IEEE Transactions on Evolutionary Computation,* 19**,** 474-489.

HASTIE, T., TIBSHIRANI, R. & FRIEDMAN, J. 2009. *The elements of statistical learning: data mining, inference, and prediction*, Springer Science & Business Media.

HE, K., ZHANG, X., REN, S. & SUN, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 770-778.

HINTON, G. E. & SALAKHUTDINOV, R. R. 2006. Reducing the dimensionality of data with neural networks. *science*, 313**,** 504-507.

HOLLANDER, M., WOLFE, D. A. & CHICKEN, E. 2013. *Nonparametric statistical methods*, John Wiley & Sons.

HORN, S. D. 1977. Goodness-of-fit tests for discrete data: a review and an application to a health impairment scale. *Biometrics***,** 237-247.

HYNDMAN, R., KANG, Y., MONTERO-MANSO, P., TALAGALA, T., WANG, E., YANG, Y. & O'HARA-WILD, M. 2019. tsfeatures: Time Series Feature Extraction. *R package version,* 1.

HYNDMAN, R., KOEHLER, A. B., ORD, J. K. & SNYDER, R. D. 2008. *Forecasting with exponential smoothing: the state space approach*, Springer Science & Business Media.

HYNDMAN, R. J. & ATHANASOPOULOS, G. 2014. *Forecasting: principles and practice*, OTexts.

HYNDMAN, R. J. & ATHANASOPOULOS, G. 2018. *Forecasting: principles and practice*, OTexts.

HYNDMAN, R. J., ATHANASOPOULOS, G., BERGMEIR, C., CACERES, G., CHHAY, L., O'HARA-WILD, M., PETROPOULOS, F., RAZBASH, S. & WANG, E. 2020. Package 'forecast'. *Online] https://cran. r-project. org/web/packages/forecast/forecast. pdf*.

HYNDMAN, R. J. & BILLAH, B. 2003. Unmasking the Theta method. *International Journal of Forecasting,* 19**,** 287-290.

HYNDMAN, R. J. & KHANDAKAR, Y. 2007. *Automatic time series for forecasting: the forecast package for R*, Monash University, Department of Econometrics and Business Statistics.

HYNDMAN, R. J., KOEHLER, A. B., SNYDER, R. D. & GROSE, S. 2002. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting,* 18**,** 439-454.

HYNDMAN, R. J., WANG, E. & LAPTEV, N. Large-scale unusual time series detection. 2015 IEEE international conference on data mining workshop (ICDMW), 2015a. IEEE, 1616-1619.

HYNDMAN, R. J., WANG, E. & LAPTEV, N. Large-scale unusual time series detection. Data Mining Workshop (ICDMW), 2015 IEEE International Conference on, 2015b. IEEE, 1616-1619.

KALCHBRENNER, N., GREFENSTETTE, E. & BLUNSOM, P. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

KALOUSIS, A. & THEOHARIS, T. 1999. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis,* 3**,** 319-337.

KANG, Y., HYNDMAN, R. J. & SMITH-MILES, K. 2017. Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting,* 33**,** 345-358.

KENDALL, M. G. & ORD, J. K. 1990. *Time-series*, Edward Arnold London.

KOLASSA, S. 2011. Combining exponential smoothing forecasts using Akaike weights. *International Journal of Forecasting,* 27**,** 238-251.

KOURENTZES, BARROW, D. & PETROPOULOS, F. 2019a. Another look at forecast selection and combination: Evidence from forecast pooling. *International Journal of Production Economics*, 209**,** 226-235.

KOURENTZES, N. 2017. Can you spot trend in time series? *Forecasting research* [Online]. [Accessed March 30.

KOURENTZES, N. 2019a. tsutils: Time Series Exploration, Modelling and Forecasting. *R package version 0.9. 0. URL: https://CRAN. R-project. org/package= tsutils*.

KOURENTZES, N. 2019b. Tsutils: Time Series Exploration, Modelling and Forecasting. *R Package Version 0.9. 0*.

KOURENTZES, N., BARROW, D. & PETROPOULOS, F. 2019b. Another look at forecast selection and combination: Evidence from forecast pooling. *International Journal of Production Economics,* 209**,** 226-235.

KOURENTZES, N. & PETROPOULOS, F. 2014. MAPA: Multiple aggregation prediction algorithm.

KOURENTZES, N., PETROPOULOS, F. & TRAPERO, J. R. 2014. Improving forecasting by estimating time series structural components across multiple frequencies. *International Journal of Forecasting,* 30**,** 291-302.

KOURENTZES, N., ROSTAMI-TABAR, B. & BARROW, D. K. 2017. Demand forecasting by temporal aggregation: using optimal or multiple aggregation levels? *Journal of Business Research,* 78**,** 1-9.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012. 1097-1105.

KÜCK, M., CRONE, S. F. & FREITAG, M. Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. 2016 International Joint Conference on Neural Networks (IJCNN), 2016a. IEEE, 1499-1506.

KÜCK, M., CRONE, S. F. & FREITAG, M. Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. Neural Networks (IJCNN), 2016 International Joint Conference on, 2016b. IEEE, 1499-1506.

KUHN, M. 2008. Building predictive models in R using the caret package. *Journal of statistical software,* 28**,** 1-26.

KUHN, M. & JOHNSON, K. *Applied predictive modeling*, Springer.

LECUN, Y. & BENGIO, Y. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks,* 3361**,** 1995.

LECUN, Y., BOTTOU, L., BENGIO, Y. & HAFFNER, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86**,** 2278-2324.

LECUN, Y., KAVUKCUOGLU, K. & FARABET, C. Convolutional networks and applications in vision. Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010. IEEE, 253-256.

LEHMAN, A. 2005. *JMP for basic univariate and multivariate statistics: a step-by-step guide*, SAS Institute.

LEMKE, C. & GABRYS, B. 2010a. Meta-learning for time series forecasting and forecast combination. *Neurocomputing,* 73**,** 2006-2016.

LEMKE, C. & GABRYS, B. Meta-learning for time series forecasting in the NN GC1 competition. Fuzzy Systems (FUZZ), 2010 IEEE International Conference on, 2010b. IEEE, 1-5.

LI, X., KANG, Y. & LI, F. 2020. Forecasting with time series imaging. *Expert Systems with Applications,* 160**,** 113680.

LIAW, A. & WIENER, M. 2002. Classification and regression by randomForest. *R news,* 2**,** 18-22.

LUNDBERG, S. M. & LEE, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems,* 30.

MA, S. & FILDES, R. 2020. Retail sales forecasting with meta-learning. *European Journal of Operational Research*.

MAKRIDAKIS, S., ANDERSEN, A., CARBONE, R., FILDES, R., HIBON, M., LEWANDOWSKI, R., NEWTON, J., PARZEN, E. & WINKLER, R. 1982. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting,* 1**,** 111-153.

MAKRIDAKIS, S. & HIBON, M. 2000. The M3-Competition: results, conclusions and implications. *International journal of forecasting,* 16**,** 451-476.

MAKRIDAKIS, S., SPILIOTIS, E. & ASSIMAKOPOULOS, V. 2018. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting,* 34**,** 802-808.

MAKRIDAKIS, S., SPILIOTIS, E. & ASSIMAKOPOULOS, V. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting,* 36**,** 54-74.

MAKRIDAKIS, S., WHEELWRIGHT, S. C. & HYNDMAN, R. J. 2008. *Forecasting methods and applications*, John wiley & sons.

MATIJAŠ, M., SUYKENS, J. A. & KRAJCAR, S. 2013. Load forecasting using a multivariate meta-learning system. *Expert systems with applications,* 40**,** 4427-4437.

MEADE, N. 2000. Evidence for the selection of forecasting methods. *Journal of forecasting,* 19**,** 515-535.

MONTERO-MANSO, P., ATHANASOPOULOS, G., HYNDMAN, R. J. & TALAGALA, T. S. 2020. FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting,* 36**,** 86-92.

NWOGU, E. C., IWUEZE, I. S. & NLEBEDIM, V. U. 2016. Some Tests for Seasonality in Time Series Data. *Journal of Modern Applied Statistical Methods,* 15**,** 382-399.

OORD, A. V. D., DIELEMAN, S., ZEN, H., SIMONYAN, K., VINYALS, O., GRAVES, A., KALCHBRENNER, N., SENIOR, A. & KAVUKCUOGLU, K. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

ORD, K., FILDES, R. A. & KOURENTZES, N. 2017. Principles of Business Forecasting. Wessex Press Publishing Co.

PEGELS, C. C. 1969. Exponential forecasting: some new variations. *Management Science***,** 311-315.

PETERS, A., HOTHORN, T. & HOTHORN, M. T. 2009. Package ‘ipred’. *0.8-7. The R Foundation for Statistical Computing*.

PETROPOULOS, F., KOURENTZES, N., NIKOLOPOULOS, K. & SIEMSEN, E. 2018. Judgmental selection of forecasting models. *Journal of Operations Management,* 60**,** 34-46.

PHILLIPS, C. L., PARR, J. M. & RISKIN, E. A. 2003. *Signals, systems, and transforms*, Prentice Hall Upper Saddle River.

POHLERT, T. 2016. Non-parametric trend tests and change-point detection. *CC BY-ND,* 4.

POHLERT, T., POHLERT, M. T. & KENDALL, S. 2016. Package ‘trend’. *Title Non-Parametric Trend Tests and Change-Point Detection*.

POLLOCK, D. S. G., GREEN, R. C. & NGUYEN, T. 1999. *Handbook of time series analysis, signal processing, and dynamics*, Elsevier.

PRIDDY, K. L. & KELLER, P. E. 2005. *Artificial neural networks: an introduction*, SPIE press.

PRUDÊNCIO, R. B. & LUDERMIR, T. B. 2004. Meta-learning approaches to selecting time series models. *Neurocomputing,* 61**,** 121-137.

PRUDÊNCIO, R. B. C., DE SOUTO, M. C. & LUDERMIR, T. B. 2011. Selecting Machine Learning Algorithms Using the Ranking Meta-Learning Approach. *Meta-Learning in Computational Intelligence*, 358**,** 225-243.

RICE, J. R. 1976. The algorithm selection problem. *Advances in computers,* 15**,** 65-118.

RICHMAN, J. S. & MOORMAN, J. R. 2000. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology,* 278**,** H2039-H2049.

SAGAERT, Y. R., AGHEZZAF, E.-H., KOURENTZES, N. & DESMET, B. 2018. Tactical sales forecasting using a very large set of macroeconomic indicators. *European Journal of Operational Research,* 264**,** 558-569.

SAID, S. E. & DICKEY, D. A. 1984. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika,* 71**,** 599-607.

SALINAS, D., FLUNKERT, V., GASTHAUS, J. & JANUSCHOWSKI, T. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting,* 36**,** 1181-1191.

SAYEMUZZAMAN, M. & JHA, M. K. 2014. Seasonal and annual precipitation time series trend analysis in North Carolina, United States. *Atmospheric Research,* 137**,** 183-194.

SHAH, C. 1997. Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting,* 13**,** 489-500.

SHANNON, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal,* 27**,** 379-423.

SIMONYAN, K. & ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

SMITH-MILES, K. A. 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR),* 41**,** 6.

SMITH, J. & WALLIS, K. F. 2009. A simple explanation of the forecast combination puzzle. *Oxford Bulletin of Economics and Statistics,* 71**,** 331-355.

SPRINGENBERG, J. T., DOSOVITSKIY, A., BROX, T. & RIEDMILLER, M. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

STONE, M. 1977. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society: Series B (Methodological),* 39**,** 44-47.

SUN, S. & FANG, C. 2017. Water use trend analysis: A non-parametric method for the environmental Kuznets curve detection. *Journal of Cleaner Production*.

SVETUNKOV, I. 2017. Statistical models underlying functions of'smooth'package for R.

SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J. & WOJNA, Z. Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 2818-2826.

TALAGALA, T. S., HYNDMAN, R. J. & ATHANASOPOULOS, G. 2018a. Meta-learning how to forecast time series. Monash University, Department of Econometrics and Business Statistics.

TALAGALA, T. S., HYNDMAN, R. J. & ATHANASOPOULOS, G. 2018b. Meta-learning how to forecast time series. *Monash Econometrics and Business Statistics Working Papers,* 6**,** 18.

TASHMAN, L. J. 2000. Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting,* 16**,** 437-450.

TEAM, R. C. 2013. R: A language and environment for statistical computing. Vienna, Austria.

THERNEAU, T., ATKINSON, B., RIPLEY, B. & RIPLEY, M. B. 2018. Package 'rpart'. *Available online: cran. ma. ic. ac. uk/web/packages/rpart/rpart. pdf (accessed on 20 April 2016)*.

TIELEMAN, T. & HINTON, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning,* 4**,** 26-31.

TIMMER, J., GANTERT, C., DEUSCHL, G. & HONERKAMP, J. 1993. Characteristics of hand tremor time series. *Biological cybernetics,* 70**,** 75-80.

VAN DEN OORD, A., DIELEMAN, S. & SCHRAUWEN, B. Deep content-based music recommendation. Advances in neural information processing systems, 2013. 2643-2651.

VANSCHOREN, J. & BLOCKEEL, H. Towards understanding learning behavior. Proceedings of the annual machine learning conference of Belgium and the Netherlands, 2006. 89-96.

VAPNIK, V. 2013. *The nature of statistical learning theory*, Springer science & business media.

VENKATACHALAM, A. & SOHL, J. E. 1999. An intelligent model selection and forecasting system. *Journal of Forecasting,* 18**,** 167-180.

VILALTA, R., GIRAUD-CARRIER, C. G., BRAZDIL, P. & SOARES, C. 2004. Using Meta-Learning to Support Data Mining. *IJCSA,* 1**,** 31-45.

VOKURKA, R. J., FLORES, B. E. & PEARCE, S. L. 1996. Automatic feature identification and graphical support in rule-based forecasting: a comparison. *International Journal of Forecasting,* 12**,** 495-512.

WANG, X., SMITH-MILES, K. & HYNDMAN, R. 2009. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing,* 72**,** 2581-2594.

WANG, Z., YAN, W. & OATES, T. 2016. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. *arXiv preprint arXiv:1611.06455*.

WELLER, M. & CRONE, S. 2012. Supply chain forecasting: Best practices & benchmarking study. *Technical Paper. Lancaster Centre for Forecasting***,** 1-42.

WIDODO, A. & BUDI, I. Feature enhancement for model selection in time series forecasting. 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2013. IEEE, 367-373.

XIA, Y. & TONG, H. 2011. Feature matching in time series modeling. *Statistical Science***,** 21-46.

YENTES, J. M., HUNT, N., SCHMID, K. K., KAIPUST, J. P., MCGRATH, D. & STERGIOU, N. 2013. The appropriate use of approximate entropy and sample entropy with short data sets. *Annals of biomedical engineering,* 41**,** 349-365.

ZHANG, D., QIAN, L., MAO, B., HUANG, C., HUANG, B. & SI, Y. 2018. A data-driven design for fault detection of wind turbines using random forests and XGboost. *IEEE Access,* 6**,** 21020-21031.

ZHAO, B., LU, H., CHEN, S., LIU, J. & WU, D. 2017. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics,* 28**,** 162-169.

ZHOU, S., LAI, K. K. & YEN, J. 2012. A dynamic meta-learning rate-based model for gold market forecasting. *Expert Systems with Applications,* 39**,** 6168-6173.

Chapter 0: