

https://dspace.org/

Time-series forecasting: an empirical evaluation of the state of the art, ensembles- and meta-learning strategies

Cawood, Nicolaas Pieter

2022
https://hdl.handle.net/10539/34570

Downloaded from a DSpace repository. https://dspace.org/

RESEARCH REPORT

Time-series Forecasting: An Empirical Evaluation of the State of the Art, Ensembles- and Meta-learning Strategies

Author:
Nicolaas Pieter CAWOOD
(2376182)

Supervisor:
Prof. Terence VAN ZYL



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

submitted to
the Faculty of Science, in fulfilment of the requirements for the degree
of
MSc (CW&RR) in Artificial Intelligence

in the

School of Computer Science and Applied Mathematics

May 6, 2022

Declaration of Authorship

I, Nicolaas Pieter CAWOOD (2376182), declare that this research report titled, "Time-series Forecasting: An Empirical Evaluation of the State of the Art, Ensembles- and Meta-learning Strategies" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this research report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this research report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the research report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

PCawood

Date: May 6, 2022

THE UNIVERSITY OF THE WITWATERSRAND

Abstract

Faculty of Science
School of Computer Science and Applied Mathematics

MSc (CW&RR) in Artificial Intelligence

Time-series Forecasting: An Empirical Evaluation of the State of the Art, Ensembles- and Meta-learning Strategies

by Nicolaas Pieter CAWOOD (2376182)

Techniques of hybridisation and ensemble learning have been popular for improving the predictive power of forecasting methods during the past two decades. With limited research that combines these two promising approaches, this study focuses on the utility of the Exponential-Smoothing-Recurrent Neural Network (ES-RNN) in the pool of base models for different ensembles. This study compares some state of the art ensemble techniques and arithmetic model averaging as a benchmark. This study experiments with the M4 forecasting competition's data set of $100,000$ time-series, and the results show that the Feature-based Forecast Model Averaging (FFORMA), on average, is the best technique for late data fusion with the ES-RNN. However, considering the M4's Daily subset of data, stacking was the only successful ensemble at dealing with the case where all base model performances are similar. The experimental results conclude that model averaging is a more robust ensemble than model selection and stacking strategies. Further, the results show that gradient boosting is superior for implementing ensemble learning strategies.

Acknowledgements

I want to express my gratitude to Spyros Makridakis and his team for organising the Markidakis forecasting competitions and making the data and submissions available. I want to thank and acknowledge the work of Slawek Smyl and Pablo Montero-Manso *et al.* for contributing their forecasting research on the ES-RNN and FFORMA. I also thank Federico Garza *et al.* for sharing their Python implementation of the FFORMA.

Finally, I thank and acknowledge supervisor, Professor Terence van Zyl for his time, encouragement and contributions towards preparing this research report.

Publications

Feature-Weighted Stacking for Nonseasonal Time Series Forecasts: A Case Study of the COVID-19 Epidemic Curves

Pieter Cawood, Terence van Zyl

2021 8th International Conference on Soft Computing & Machine Intelligence (ISCMI) | 2021-11-26 | Conference article | DOI: 10.1109/iscmi53840.2021.9654809

Evaluating the State of the Art, Forecasting Ensembles- and Meta-learning Strategies for Model Fusion

Pieter Cawood, Terence van Zyl

2022-02 | Research article | DOI: 10.48550/arXiv.2203.03279

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

Publications iv

1 Introduction 1
- 1.1 Background . . . 3
- 1.2 Definitions . . . 5
- 1.3 Problem Statement . . . 6
- 1.4 Significance and Motivation . . . 7
  - 1.4.1 Research Aims . . . 7
  - 1.4.2 Objectives . . . 8
- 1.5 Research Questions . . . 8
- 1.6 Delineations, Limitations and Assumptions . . . 9
- 1.7 Outline . . . 9

2 Overview of the Base Models 10
- 2.0.1 Seasonal ARIMA . . . 10
  - The ARIMA Sub-Processes . . . 11
  - Auto ARIMA . . . 11
  - Preprocessing . . . 12
- 2.0.2 Theta . . . 12
  - Theta's Forecasting Procedure . . . 12
  - Preprocessing . . . 12
- 2.0.3 Damped Exponential Smoothing . . . 13
  - Preprocessing . . . 13
- 2.0.4 Comb . . . 13
  - Single Exponential Smoothing (SES) . . . 13
  - Holt-Winters (HW) . . . 14
  - Preprocessing . . . 15
- 2.0.5 The ES-RNN . . . 15
  - Preprocessing . . . 16
  - Forecasts by the ANN . . . 17
  - The ANN Architectures . . . 18
  - Loss Function and Optimiser . . . 19

3 Research Methodology **20**
3.1 Research design . . . 20
3.2 The M4 Forecasting Competition . . . 20
3.2.1 Naïve model 2 . . . 20
3.2.2 Performance Metrics . . . 21
3.2.3 Data Description . . . 22
Forecasting Horizons . . . 23
3.3 Overview of Ensembles . . . 23
3.3.1 Simple Model Averaging (AVG) . . . 23
3.3.2 The FFORMA . . . 23
Meta-learning Model . . . 24
Forecasts Combination . . . 25
Meta-features . . . 25
3.3.3 The FFORMS . . . 26
3.3.4 Feature-weighted Neural Stacking (NN-STACK) . . . 26
Meta-learning Model . . . 27
Meta-features . . . 27
3.3.5 Feature-weighted Neural Averaging (NN-AVG) . . . 27
Meta-learning Model . . . 28
Forecasts Combination . . . 28
Meta-features . . . 28
3.4 Overview of Neural Basis Expansion Analysis (N-BEATS) . . . 28
3.4.1 Building Blocks . . . 29
The Generic Architecture . . . 30
Architecture to Model Trend . . . 30
Architecture to Model Seasonality . . . 31
3.4.2 Doubly Residual Stacking . . . 31
3.4.3 Ensembling . . . 32
3.5 Details of Experimentation . . . 32
3.5.1 The Evaluation Protocol . . . 32
3.5.2 Research Instruments . . . 32
Hardware and Software . . . 32
Data . . . 33
Third-party Libraries . . . 33
Preprocessing of the Ensembles' Input Data . . . 34
The Ensembles' Architectures, Hyperparameters and Training . . . 34
3.5.3 Results Data and Analysis . . . 36
3.6 Limitations . . . 37
3.7 Ethical Considerations . . . 37

4 Results and Discussion **38**
4.1 Detailed Observations . . . 38
4.1.1 Hourly Subset . . . 39
4.1.2 Weekly Subset . . . 40
4.1.3 Daily Subset . . . 41
4.1.4 Monthly Subset . . . 43

4.1.5 Yearly Subset 44
4.1.6 Quarterly Subset 44
4.1.7 Overall Results 45

5 Conclusion **47**

A The FFORMA algorithm **49**

B The FFORMA meta-features **50**

Bibliography **51**

List of Figures

3.1 The proposed FFORMA forecasting pipeline. 26

4.1 Hourly subset OWA distributions 39
4.2 Weekly subset OWA distributions 40
4.3 Daily subset OWA distributions 41
4.4 Monthly subset OWA distributions 42
4.5 Yearly subset OWA distributions 43
4.6 Quarterly subset OWA distributions 44

List of Tables

3.1 Number of series per data frequency and domain [19]. 22
3.2 The FFORMA's hyperparameters. 35
3.3 The NN-STACK hyperparameters. 36
3.4 The NN-AVG hyperparameters. 36

4.1 Average OWA performance over fifty test-folds of the M4 data set. N-BEATS† reproduced here for comparison. 39

B.1 Features used in the FFORMA framework [15]. 50

List of Abbreviations

ABM	**A**gent-**B**ased **M**odelling
AR	**A**uto **R**egressive
ARIMA	**A**utoregressive **I**ntegrated **M**oving **A**verage
AVG	Model Averaging
BMA	**B**ayesian **M**odel **A**veraging
ES	**E**xponential **S**moothing
ES-RNN	**E**xponential-**S**moothing-**R**ecurrent **N**eural **N**etworks
FFORMA	**F**eature-based **For**ecast **M**odel **A**veraging
GBM	**G**radient **B**oosting **M**achines
HW	**H**olt **W**inters
LVP	**L**evel **V**ariability **P**enalty
LSTM	**L**ong **S**hort-**T**erm **M**emory
MA	**M**oving **A**verage
MAE	**M**ean **A**bsolute **E**rror
MASE	**M**ean **A**bsolute **S**caled **E**rror
ML	**M**achine **L**earning
MLE	**M**aximum-**L**ikelihood **E**stimation
MLP	**M**ulti**l**ayer **P**erceptron
MSIS	**M**ean **S**caled **I**nterval **S**core
N-BEATS	**N**eural-**B**asis **E**xpansion **A**nalysis
NN-AVG	**N**eural **N**etwork Model Averaging
NN-STACK	**N**eural **N**etwork Model Stacking
OWA	**O**verall **W**eighted **A**verage
PF	**P**oint **F**orecasts
PI	**P**rediction **I**ntervals
QR	**Q**uantile **R**egression
RNN	**R**ecurrent **N**eural **N**etwork
RF	**R**andom **F**orest
SES	**S**ingle **E**xponential **S**moothing
SGD	**S**tochastic **G**radient **D**escent
SIR	**S**usceptible **I**nfected, and **R**ecovered
SMA	**S**imple **M**odel **A**veraging
XGB	e**X**treme **G**radient **B**oosting

List of Symbols

$\vec{x}$	Time-series
n	Number of time-series observations
h	Forecasting horizon
$\vec{w}_M$	Weightings for all models of an ensemble
$\vec{F}$	Functions that extract the meta-features (f)
$\vec{M}$	Time-series forecasting models (m)
H	Hessian function
G	Gradient function
L	Loss function
ρ	Spearman's rank correlation coefficient

Chapter 1

Introduction

Although it is a difficult task, forecasting time-series is nevertheless an important task that boasts numerous research efforts. Markidakis *et al.* [1] emphasise two facts about the field: first, no one has prophetic powers to predict the future accurately, and second, all predictions are subject to uncertainty, especially within social contexts.

FitzRoy first coined the term "forecasts" nearly two hundred years ago, a meteorologist who made daily weather predictions after considering the value of predicting coming storms might hold for sailors at sea. The prediction of storms fueled the growing realisation that weather does follow predictable patterns over large areas [2]. The potential of forecasting was subsequently realised for several fields within the physical sciences; moreover, our understanding of the sciences grew, along with increased access to large amounts of information. In the nineteenth century, Yule [3] made a significant contribution by proposing that every time series can be thought of as the realisation of a stochastic process. This very concept led to the rise of several statistical time-series techniques, including the autoregressive (AR) and moving average (MA) models [4]. Nowadays, most forecasting literature implements artificial neural networks (ANN) or combines them with traditional models. Zhang *et al.* [5] identify this attraction to ANNs as their ability to self-adapt and model underlying relationships within the data, even when the relationships are unknown.

The Makridakis Competitions [1] (or M-Competitions) are undoubtedly the most influential initiative driving continuous research in improved forecasting methods. The M-competitions are a standard benchmark for establishing innovatory forecasting methods [1], [6]. Markidakis *et al.* [1] point out that machine learning (ML) models have delivered poor performances for the forecasting of social sciences when considering ML performance in the M-competitions prior to the recent M5 competition. The M5 competition was the first in the series of M-competitions with hierarchical time-series data [6]. Subsequently, the M5 competition led research efforts to confirm the dominance of ML methods in forecasting homogeneous data.

There exist many techniques underpinned by different principles to perform forecasting, and numerous forecasting studies have shown that combination models generally outperform individual ones [7]–[9]. Given the top entries of the M5 competition, methods that adopt ensemble learning and gradient boosting produce state of the art forecasting results [10]. It is no surprise that ensembles were used in the three winning submissions of the M5. Ensembles are known to outperform their constituent models for many machine learning problems [11]. Likewise, integrating models with different architectures has been reported to deliver superior results [12], [13]. For this reason, this study is focused on combination strategies rather than individual forecast methods, and the reader interested in a comprehensive study of statistical and ML forecast models are referred to a large comparative study by Makridakis *et al.* [14].

From a review of contemporary univariate time-series forecasting literature, the M4 competition's data set was noted as the most prominently used, and thus the most relevant data set for this study [15]–[17]. The data is comprised of $100,000$ time-series of different seasonality from the micro, macro, industry, finance, demographic and other domains.

[1] https://mofc.unic.ac.cy/

1.1 Background

Data science/ML techniques are currently the preferred family for research of forecasting time-series, owing to their predictive accuracy [18]. Makridakis *et al.* [19] recently presented a large scale comparative study of 100,000 time-series and 61 forecasting methods. They compare the accuracy of statistical methods (e.g., Autoregressive integrated moving average (ARIMA), Holt-Winters (HW), and others) versus that of ML ones (e.g., Multilayer Perceptron (MLP), recurrent neural network (RNN) and others.) Furthermore, similar to others, they find that utilising models of both kinds in combination methods produce the best results [7]–[9].

Despite the immense diversity of the forecasting models, the no free lunch theorem holds that no single algorithm universally outperforms any other for all problems [20]. Considering the theorem, an essential question is raised about choosing the best algorithm for a specific time-series. Arinze [21] first introduced meta-learning to select one forecasting model among six others based on their learnt performance for six time-series statistics (e.g., autocorrelation, trend, coefficients.) Raftery *et al.* [22] point out that single model selection is open to issues of model uncertainty and proposed using Bayesian model averaging (BMA) to make weighted average predictions as the posterior probability of each model being correct given the same data set. Stacking is another ensemble method that uses a combiner model to learn from the predictions of multiple methods using cross-validation, instead of using posterior probabilities [23]. Clarke [23] reported stacking as a more robust method than BMA for computations that involve sensitive changes over their variables. Ribeiro and Coelhoa [24] studied contemporary ensemble methods in forecasting time-series within agribusiness. In their work, they compare approaches of bagging with random forests (RF), boosting with gradient boosting machines (GBM), extreme gradient boosting machine (XGB)

and a stacked generalisation of machine learning models. Ribeiro and Coelhoa's findings suggest that gradient boosting methods generally produce the lowest prediction errors.

Ensemble methods' performance might also be improved by utilising additional time-series statistics as meta-features in the learning process. Cawood and van Zyl [25], for example, proposed a feature-weighted stacking approach that improves the regression from weak learner predictions by supplementing them with time-series statistics extracted from the input series. Recent work has established many valuable statistics for meta-learning, including, but not limited to, the linearity, curvature, autocorrelation, stability and entropy [15], [26], [27].

The top two submissions of the M4 forecasting competition both implement ensemble learning to produce state of the art results. The runner-up: FFORMA framework implements an XGB meta-learning model that learns the model weightings of the ensemble based on their performance learned for several meta-features extracted from the input series [15]. The weightings previously learned by the trained meta-model are then used to combine the models' predictions as a feature-weighted average.

The Exponential Smoothing-Recurrent Neural Network (ES-RNN) is a hybrid forecasting model and winning submission of the M4 competition that merges a modified Holt-Winters and dilated long short-term memory (LSTM) stacks [16]. Hybrid approaches combine linear and nonlinear models since time series are rarely pure linear or nonlinear in reality [28]. Hybrid forecasting models was first proposed by Zhang [28], who showed that the hybridisation of the ARIMA and MLP models yield improved accuracy since it takes advantage of combining the strength of both linear- and nonlinear models. The authors note that the main advantage of neural networks is their ability to do nonlinear modelling. In their implementation, the MLP learns from the residuals of the linear ARIMA method to make the final integrated

predictions. Hybrid methods have gained traction in recent time-series forecasting research [29]–[31]. The most common approach is to combine deep learning with a linear method, similar to the initial work by Zhang. Some have proposed hybridising fuzzy logic techniques with ML models [32], [33]. Others have also reported success for integrating ML models with genetic algorithms like particle swarm- and ant colony optimisation [34], [35]. As it stands, the Neural Basis Expansion Analysis (N-BEATS) achieves state of the art forecasting accuracy by 3% over the ES-RNN [17]. N-BEATS is a hybrid forecasting method that integrates a fully connected neural network with traditional time-series decomposition. N-BEATS proposes a novel doubly residual topology that stacks blocks of different architectures to model the different time-series components. The N-BEATS stacking topology resembles the DenseNet architecture proposed by Huang *et al.* [36]; however, it has two residual branches that run over both foreword and backward prediction branches of each layer. The neural network thus produces expansion coefficients that are projected to the block basis functions in each layer to produce partial forecasts and backcasts that assist downstream blocks to ignore components of their inputs that are not useful.

1.2 Definitions

The following key terms are defined within the context of this research, and they are intended to help the reader understand the study better.

Base model/weak learner: A single statistical or machine learning forecast method.

Characteristic statistics/meta-features: Endogenous features extracted from the input series that statistically describe the complexity and behaviour.

Ensemble learning: The combination of multiple algorithms' predictions to obtain better predictive performance.

Hybridisation/model integration: The integration of two or more distinctively powerful techniques to blend their unique advantages to deliver a single- and more powerful method that encapsulates the advantages of each of its constituent models.

Meta-learner/strong learner: The core learner of the ensemble that learns from both the metadata and the predictions of two or more base models.

Stacking: A stacked generalisation of base models that combines their predictions.

1.3 Problem Statement

Techniques of model integration and ensembling have improved forecast models' predictive power in the literature. Consequently, the combination of these two concepts has delivered state of the art results, as noted for the winning submission for the M4 competition: the ES-RNN. Furthermore, given the competition ranking of the FFORMA framework, the adoption of a feature weighted meta-learning strategy has shown to be the most robust combination technique amongst numerous others.

However, the pool of base models for the FFORMA's implementation were traditional statistical models. No study researches how well this ensemble would perform when an advanced model like the ES-RNN is added to the pool of base models. Furthermore, there is a gap in the literature that evaluates the performance of these techniques against other ensembles, using the forecasts from the same pool of forecasting models. To this end, this study implements and evaluates different forecast models and ensembles to

empirically determine whether any of them is a state of the art ensembling technique.

1.4 Significance and Motivation

Makridakis *et al.*[1] pointed out that machine learning models combined with traditional time-series forecast approaches perform exceptionally well. The dominance of hybrid models was evident after considering the performance of the ES-RNN model in the M4 competition. However, research also showed that a combination of models generally improved standalone forecast models' predictions. Therefore, it appeared that investigating different ensembling techniques that include hybrid machine learning base models requires further investigation.

This study proposed to use the forecasts of selected submissions of the M4 forecasting competition as weak learner forecasts for different ensemble techniques (i.e., model selection, model averaging and stacking). The findings provide future researchers with empirical evidence of a state of the art benchmark for their work. Further, the study reports concrete evidence that ensemble learning should always be utilised to boost the accuracy of time-series forecast methods.

1.4.1 Research Aims

The stochastic environments of social contexts make the forecasting thereof difficult. Nevertheless, maximising the accuracy of forecasting models is of great importance as they offer countless benefits. These benefits are sought after in industries, so it is typical to find forecasting literature related to business implicated time-series [37], [38].

The study aimed to combine state of the art ensemble techniques for data and model fusion to forecast time-series. Furthermore, the study aimed to

provide a comparison of the results with that of the top forecasting methods to answer whether a single method might be singled out as the state of the art approach to forecast univariate time-series.

1.4.2 Objectives

The objectives of this study were as follows:

- To survey the methodologies of the top-two submissions for the M4 forecasting competition and analyse how to combine their key advantages of ensembling and hybridisation.
- To evaluate the superiority of ensembles in forecasting and compare the performance of the ES-RNN and N-BEATS against other ensemble techniques.
- Another objective was to deliver a study that compares multiple ensembles using the same data produced by a shared pool of forecasting models.
- The primary objective was to determine whether any method might be singled out as the state of the art forecasting technique for future research efforts.

1.5 Research Questions

The questions this study proposes to answer are as follows:

1. How does the accuracy of state of the art forecasting methods, compare to ensembles that use the ES-RNN as a weak learner?
2. How does the accuracy of different ensemble approaches (model stacking, model averaging and model selection) compare against each other?

1.6 Delineations, Limitations and Assumptions

Although the M4 competition has an optional prediction interval forecasting component, this study is limited to point forecasts only. This study only considers the M4 competition's data set as it has the best potential for comparing the results with the literature. As a result, the outcomes might not apply to other data sets or problems other than time-series forecasting. Lastly, the study does not implement the base models discussed but instead reuse the original forecasts from the M4 submissions as the data for the ensembles.

1.7 Outline

The remainder of this report is organised as follows: Chapter 3 proposes the methodology that this study follows. First, the M4 data set and requirements are discussed. The study then overviews the base models and ensembling methodologies. The chapter's final sections report the experiment's evaluation protocol and other essential research instruments. In Chapter 4, the results are discussed, and finally, Chapter 5 concludes the findings of this study and suggestions are made for future research efforts.

Chapter 2

Overview of the Base Models

The combination of traditional statistical models and deep learning models have been shown to produce promising results [25]. Consequently, this study chose four statistical methods, among which two of them were the best methods for previous Makridakis forecasting competitions. Lastly, this study experiments with the ES-RNN (a hybrid deep learning model) and the winner of the M4 forecasting competition. This section gives an overview of the background for base each model used in the ensembling experimentation.

2.0.1 Seasonal ARIMA

The ARIMA method is a standard method for comparing forecast methods' performances. Fattah *et al.* [39] describe the ARIMA as an iterative approach with three sub-processes. The seasonal ARIMA model might be written as follows:

$$\text{ARIMA}(p, d, q)(P, D, Q)_m \tag{2.1}$$

where p denotes the number of autoregressive terms, d denotes the number of differences, and q is the number of moving averages. The seasonal terms (P, D and Q) are similar to the nonseasonal ones, but the timesteps are defined by m- the frequency of the seasonality.

The ARIMA Sub-Processes

The autoregressive process assumes that the time-series is linear and might be modelled with the linear function:

$$Y_t = \alpha_1 Y_{t-1} + \epsilon_t \tag{2.2}$$

where α_1 is the self-regression coefficient and ϵ_t is a random shock component on each observation.

The integrated process represents the differencing of observations to make the time-series stationary. (i.e., observations are replaced with the difference between themselves with previous observations.) The integrated process is simply defined as:

$$Y_t = Y_{t-1} + \epsilon_t \tag{2.3}$$

where ϵ is white noise in the signal.

The moving average process models the past error and might be found using the following equation:

$$Y_t = \epsilon_t - \theta_1 \epsilon_{t-1} \tag{2.4}$$

where θ_1 is the coefficient to be multiplied by the previous error.

Auto ARIMA

This implementation automates the selection of the ARIMA parameters using the auto-ARIMA method that uses maximum-likelihood estimation (MLE) to approximate the ARIMA parameters [40].

Preprocessing

ARIMA is intended to model stationary time-series, and thus it is common to remove the trend and stabilise the variance as before the model is fit.

2.0.2 Theta

The Theta model was the best method of the M3 forecasting competition [41]. Theta is a simple forecasting method that averages the extrapolated Theta-lines, computed from two given Theta-coefficients, applied to the second differences of the time-series:

$$X''_{data} = X_t - 2X_{t-1} + X_{t-2} \tag{2.5}$$

$$X''_{new} = \theta \cdot X''_{data} \tag{2.6}$$

The first Theta-line ($\theta = 0$) describes the time-series through a linear trend, and it is extrapolated using linear regression. The second Theta-line (usually $\theta = 2$) is extrapolated using simple exponent smoothing (SES) [42].

Theta's Forecasting Procedure

The time-series are firstly decomposed into the two Theta-lines using equation 2.6. The extrapolation procedures mentioned above are then applied, and their forecasts are averaged to produce a combined forecast.

Preprocessing

A 90% autocorrelation test is adopted to determine whether the data are seasonal, and when it is the case, the data are seasonally adjusted by applying classical multiplicative decomposition.

2.0.3 Damped Exponential Smoothing

Holt's [43] linear trend method extended exponential smoothing to forecast data that includes a trend component. McKenzie and Gardner [44] describes the Damped method as a modification of Holt's method that adds an autoregressive-damping parameter ϕ imposed on the trend component.

The model is thus found using the following additive equations:

$$y_t = l_{t-1} + \phi b_{t-1} + \epsilon_t \tag{2.7}$$

$$l_t = l_{t-1} + \phi b_{t-1} + (1-\alpha)\epsilon_t \tag{2.8}$$

$$b_t = \phi b_{t-1} + (1-\beta)\epsilon_t \tag{2.9}$$

where y_t is the time-series observed values, l_t denotes the series' level, b_t is the gradient of the linear trend and ϵ_t is a single source of error.

Preprocessing

The same preprocessing is applied as per the Theta model.

2.0.4 Comb

Comb (or COMB S-H-D) is the arithmetic average of the three exponential smoothing methods: Single, Holt-Winters and Damped exponential smoothing [45]. Comb was the winning approach for the M2 forecasting competition, and it was used as a benchmark for evaluating all other methods in the M4 forecasting competition.

Single Exponential Smoothing (SES)

Nazim and Afthanorhan [46] mentioned that SES is used to forecast nonseasonal time-series without a trend. They give the general equation for SES

as:

$$F_{t+m} = \alpha y_t + (1 - \alpha) F_t \tag{2.10}$$

where F_{t+m} is the single exponential smoothed value at the m-step-ahead period, y_t is the actual value at time step t, α is a smoothing constant between 0 and 1, and F_t is the forecast at time step t.

Holt-Winters (HW)

In contrast to SES, the HW exponential smoothing was proposed for seasonal data that exhibits a trend [47]. Kalekar [47] mentions that there are two HW models for different types of seasonality, namely the multiplicative- and additive seasonal model.

The Multiplicative Seasonal Model is appropriate for time-series with a seasonal pattern, with an amplitude proportional to the average level of the series. The time-series is modelled using the following notation:

$$y_t = (b_1 + b_2 t) S_t + \epsilon_t \tag{2.11}$$

where b_1 is the base signal, b_2 is an optional linear trend component, S_t is a multiplicative seasonal factor, and ϵ_t is the random error component.

The Additive Seasonal Model is appropriate for time-series with a seasonal pattern, with an amplitude independent of the average level of the series. The time-series is modelled using the following notation:

$$y_t = b_1 + b_2 t + S_t + \epsilon_t \tag{2.12}$$

where b_1 is the base signal, b_2 is an optional linear trend component, S_t is a multiplicative seasonal factor, and ϵ_t is the random error component.

Preprocessing

Seasonal adjustments are taken into account according to the Theta model.

2.0.5 The ES-RNN

The ES-RNN method fuses ES models with LSTM networks to produce more accurate forecasts than those predicted by either statistical or ML approaches. Smyl [16] describes the three main elements of the ES-RNN approach as deseasonalisation and adaptive normalisation, generation of forecasts and ensembling.

The seasonality patterns of the time-series provided for the M4 competition are diversified as they are in the format of numeric vectors without timestamps, and they originated from various sources. The first element of Smyl's approach normalises and deseasonalises the series on the fly by ES decomposition. The ES decomposition also computes the level, seasonality, and second seasonality components to integrate them with the RNN forecasts.

Smyl's methodology allocates the predictions of the time-series trends to the RNN method since it can model nonlinear relationships. It is common for hybrid forecasting models to exploit the benefits of linear and nonlinear methods by integrating them [28], [48]. The second element is a dilated LSTM network, which takes the deseasonalised and normalised data to discover the potential nonlinear relationships within the data [49].

The final element is the ensembling of the forecasts from multiple instances of the hybrid model achieved by the first two steps. The ensembling helps mitigate the model and parameter uncertainty [50] and provides the

advantage of averaging over model combinations [51]. In Smyl's methodology, the hybrid instances are trained with the same architecture for independent randomly initialised runs and different subsets of the time-series if computationally feasible. The final forecast for a given series is the average of the forecasts produced by the top-N best models.

Preprocessing

Since the M4 competition's data are from various sources, it contains different seasonality patterns, even when categorised under the same frequency.

In Smyl's methodology, deseasonalisation is achieved on the fly using ES models for time-series that are nonseasonal (yearly and daily frequencies), single-seasonal (monthly, quarterly and weekly frequencies) and double-seasonal (hourly frequency) [52]. In addition, the formulas are updated to remove the linear trend component, which is modelled using the ANN instead. The updated formulas are as follows:

Nonseasonal models:

$$l_t = \alpha y_t + (1-\alpha) l_{t-1} \tag{2.13}$$

Single-seasonality models:

$$\begin{aligned} l_t &= \alpha y_t / s_t + (1-\alpha) l_{t-1} \\ s_{t+K} &= \beta y_t / l_t + (1-\beta) s_t \end{aligned} \tag{2.14}$$

Double-seasonality models:

$$\begin{aligned} l_t &= \alpha y_t / (s_t u_t) + (1-\alpha) l_{t-1} \\ s_{t+K} &= \beta y_t / (l_t u_t) + (1-\beta) s_t \\ u_{t+L} &= \gamma y_t / (l_t s_t) + (1-\gamma) u_t \end{aligned} \tag{2.15}$$

where y_t is the value of the series at time step t; l_t, s_t and u_t are the level, seasonality and second-seasonality components, respectively; K denotes the number of seasonal observations (i.e., four for quarterly, 12 for monthly and 52 for weekly) and L is the number of double-seasonal observations (168 for the hourly frequency data.) The parameters α, β and γ are smoothing factors that control the influence of the series' level, trend and seasonality components.

Smyl adopts normalisation using the typical approach of constant size, rolling input and output windows to normalise the level and seasonality components produced by the ES methods in 2.13 - 2.15. The input window size is defined after experimentation, and the output window size is always equal to the length of the forecasting horizon. The values of the input and output windows are then divided by the last value of the level of the input window and, if the series is seasonal, additionally divided by the seasonality component. Lastly, the log function is applied to counter the effects of outliers on the learning process, and the values are thus squashed to the logarithmic scale.

Furthermore, the time-series' domain (e.g., micro, macro and finance) are one-hot encoded and presented as the only meta-features to the RNN model.

Forecasts by the ANN

The normalised, deseasononalised and squashed values of level and seasonality and the meta-features are presented as inputs to the ANN. The outputs of the ANN are then integrated to complete the ES-RNN hybridisation in the following way:

Nonseasonal models:

$$\hat{y}_{t+1,\ldots,t+h} = \exp\left(NN(x)\right) \times l_t \tag{2.16}$$

Single-seasonal models:

$$\hat{y}_{t+1,\ldots,t+h} = \exp\left(NN(x)\right) \times l_t \times s_{t+1,\ldots,t+h} \tag{2.17}$$

Double-seasonal models:

$$\hat{y}_{t+1,\ldots,t+h} = \exp\left(NN(x)\right) \times l_t \times s_{t+1,\ldots,t+h} \times u_{t+1,\ldots,t+h} \tag{2.18}$$

where $NN(x)$ models the linear trend component from the input **x**, a vector of the preprocessed data discussed above. The l, s and u terms are outputs of the ES models during the preprocessing step and h denotes the forecast horizon.

The ANN Architectures

Although the ES-RNN's architectures are different for each of the six data subsets, this study reviews some of their key components.

The ANNs of the ES-RNN are constructed from dilated LSTM [49] stacks, and in some cases, followed by a nonlinear layer and always a final linear layer. Smyl refers to the linear layer as the "adapter" layer since it adapts the size of the last layer to the size of the forecasting horizon, or twice the size of the forecasting horizon for prediction interval (PI) models.

DilatedRNNs improve regular RNNs' performance with significantly fewer parameters, owing to the dilated recurrent skip connections [49]. Smyl also extends the dilated LSTM with an attention mechanism that exposes the hidden states to the weights of the previous states- a horizon equal to the dilation. To achieve this, Smyl embeds a linear two-layer network into the LSTM.

Loss Function and Optimiser

The ES-RNN implements a pinball loss function to fit the models using Stochastic Gradient Descent (SGD.) The loss is defined as follows:

$$\begin{aligned} L_t &= (y_t - \hat{y}_t)\tau, \text{ if } y_t \geq \hat{y}_t \\ &= (\hat{y}_t - y_t)(1 - \tau), \text{ if } \hat{y}_t > y_t \end{aligned} \tag{2.19}$$

where τ is configured typically between 0.45 and 0.49. Smyl notes that the pinball function is asymmetric, and that it penalises values outside a quantile range differently to deal with any biasing, and minimising it produces quantile regression.

A level variability penalty (LVP) is implemented as a regulariser to smooth the level values in the loss function. Smyl notes that this drastically improves the performance of the ANN as it can concentrate on modelling the trend instead of over-fitting seasonality-related patterns. The ES-RNN's LVP is found as follows:

1. Compute the log change, i.e., $d_t = \log(y_{t+1}/y_t)$
2. Compute the difference of the changes: $e_t = d_{t+1} - d_t$
3. Square and average the differences

Lastly, the LVP is multiplied by a constant parameter in the range of 50 - 100 before adding it to the loss functions.

Chapter 3

Research Methodology

3.1 Research design

The study undertook a confirmatory experimental research design to experimentally determine how effective some ensembles are. The study reused the forecasts of some of the M4 competition submissions as the inputs of the ensembles to determine by how much the accuracy of the ES-RNN might have been boosted with an ensemble learning strategy. Eventually, the study tests and supports the hypothesis that a feature-weighted ensemble provides state of the art forecasting performance.

3.2 The M4 Forecasting Competition

3.2.1 Naïve model 2

The Naïve model 2 is a statistical benchmark of the M4 Competition. The accuracy metric for the M4 competition divides the accuracy for a given method by the accuracy of the Naïve model 2. The M4 accuracy metric thus presents scores relative to the simple Naïve model 2. Chen *et al.* [53] describe how the Naïve model 2 produces forecasts: the forecast at time step t is obtained by multiplying the value at the previous time step with the growth rate between the previous two values. The Naïve model 2 forecasts are found

by the following equation:

$$\hat{Y}_t = Y_{t-1}[1 + \frac{Y_{t-1} - Y_{t-2}}{Y_{t-2}}] \tag{3.1}$$

where Y_t denotes the actual value of the time-series at time step t and $\hat{Y}_t$ is the predicted value of Y_t.

3.2.2 Performance Metrics

The competition rules are available on the M4 competition's website [1]. The participants had to submit point forecasts (PFs) and, optionally, prediction intervals (PIs) for 100,000 series.

The overall weighted average (OWA) computes the average of the two most popular accuracy measures to evaluate the performance of the PFs relative to the Naïve model 2. The OWA, therefore, combines the two metrics of the symmetric mean absolute percentage error (sMAPE) [54] and mean absolute scaled error (MASE) [55], each calculated as follows:

$$sMAPE = \frac{1}{h}\sum_{t=1}^{h}\frac{2|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|} \times 100\% \tag{3.2}$$

$$MASE = \frac{1}{h}\frac{\sum_{t=n+1}^{n+h}|Y_t - \hat{Y}_t|}{\frac{1}{n-m}\sum_{t=m+1}^{n}|Y_t - Y_{t-m}|} \tag{3.3}$$

where Y_t is the actual time-series value at time step t, $\hat{Y}_t$ is the predicted value of Y_t and h the length of the forecasting horizon. The denominator and scaling factor of the MASE formula is the in-sample MAE from one-step-ahead predictions of the Naïve model 2. Where n is the number of data points. The term m defines the time interval between each successive observation, i.e., 12 for time-series that have a monthly frequency, four for those

[1] https://usermanual.wiki/Document/M4CompetitorsGuide.1491768831/html

with a quarterly frequency, 24 for hourly series and one for the other frequencies that are nonseasonal series. The OWA error is then computed as the average of the MASE and sMAPE errors relative to the Naïve model 2 predictions as follows:

$$OWA = \frac{1}{2}\left(\frac{MASE}{MASE_{Naive_2}} + \frac{sMAPE}{sMAPE_{Naive_2}}\right) \tag{3.4}$$

3.2.3 Data Description

The M4 data set contains 100,000 time-series of different frequencies, i.e., yearly, quarterly, monthly, weekly, daily and hourly. The minimum number of observations varies for the different subsets, e.g., 13 for the yearly series and 16 for quarterly. The data is freely available on Github [2] and Table 3.1 provides a summary of the number of series per frequency and domain. Economic, Finance, Demographics, and Industries are among the domains, with data from Tourism, Trade, Labor and Wage, Real Estate, Transportation, Natural Resources, and the Environment also included.

TABLE 3.1: Number of series per data frequency and domain [19].

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

Redd *et al.* [56] describe the data as univariate, with no temporal links between the series, making it difficult to forecast because the series is the only representation of the signal.

[2] https://github.com/Mcompetitions/M4-methods/tree/master/Dataset

Forecasting Horizons

The length of the forecasts for each frequency are six forecasts for yearly data, eight forecasts for quarterly data, 18 forecasts for monthly data, 13 forecasts for weekly data, 14 forecasts for daily data, and 48 forecasts for hourly data.

3.3 Overview of Ensembles

This study experiments with three different kinds of ensemble techniques, namely, model stacking, model averaging and model selection. This section gives an overview of each ensembling technique used in the experiments that are different in their complexity and meta-learning strategies.

3.3.1 Simple Model Averaging (AVG)

As a simple benchmark, this study implements a model averaging technique that averages the forecasts of the weak learners [57]. The combined forecast for a set of forecast models M, might thus be expressed by the following notation:

$$\hat{y}_t = \frac{1}{n} \cdot \sum_{m=1}^{n} y_{m_t} \tag{3.5}$$

where n is the number of models in M and y_{m_t} is the forecast of a single model at time step t.

3.3.2 The FFORMA

The FFORMA framework adopts a feature-weighted model averaging strategy [15]. A meta-learner learns how effectively a pool of forecasting models are at their task for different regions of the meta-feature space and then

combines their predictions based on the learned model weightings. In addition, the FFORMA implements the gradient tree boosting model of XGBoost, which the authors note is computationally efficient and has shown promising results for other problems [58].

Montero-Manso *et al.*'s [15] meta-learning methodology use inputs of nine models and features extracted from the time-series, which measures the characteristics of a time-series: including, but not limited to, features of lag, correlation, the strength of seasonality and spectral entropy.

Meta-learning Model

Montero-Manso *et al.* [15] implement a custom objective function for the XGBoost to minimise. XGBoost requires both a gradient and hessian of the objective function to fit the model. The functions for their model is derived as follows.

The term $p_m(f_n)$ is firstly defined as the output of the meta-learner for model m. A soft-max transformation is applied to the numeric values to compute the model weights as the probability that each model is the best as follows:

$$w_m(f_n) = \frac{\exp\left(p_m(f_n)\right)}{\sum_m \exp\left(p_m(f_n)\right)} \tag{3.6}$$

where $w_m(f_n)$ is the weight produced by the GB meta-learner for base model $m \in M$. For each n time-series, the contribution of each base model m towards the OWA error is denoted as L_{nm}. The product of each base model's weight and loss are summed to produce the weighted average loss function as follows:

$$\bar{L}_n = \sum_{m=1}^{M} w_m(f_n) L_{nm} \tag{3.7}$$

where $\bar{L}_n$ is the weighted average loss over the set of base models M. The gradient of $\bar{L}_n$ is then computed as follows:

$$G_{nm} = \frac{\partial \bar{L}_n}{\partial p_m(f_n)} = w_{nm}(L_{nm} - \bar{L}_n) \tag{3.8}$$

The hessian of the objective function is then finally derived as follows:

$$H_{nm} = \frac{\partial G_n}{\partial p_m(f_n)} \approx w_{nm}(L_{nm}(1 - w_{nm}) - G_{nm}) \tag{3.9}$$

In order to minimise the objective function $\bar{L}$, the functions G and H are passed to XGBoost, and the model's hyperparameters are found using Bayesian optimisation on a limited search space that is determined based on preliminary results and rules-of-thumb.

Forecasts Combination

Montero-Manso *et al.*'s [15] methodology combines the predictions of their pool of forecasting models using algorithm 1 (see Appendix A). The forecasts are produced from the meta-learner's estimated model weightings, given the meta-features of a new time-series. Thus, the combination is achieved by using $w(f_x)$, an M-vector of weights, with the forecasts of a set of M forecasting models for each time-series x. Figure 3.1 depicts this study's proposed FFORMA forecasting pipeline.

Meta-features

Each time-series feature is computed by a function f. The FFORMA extracts 43 meta-features from each time-series using the R package called *tsfeatures* [59]. Table B.1 (see Appendix B) provides brief descriptions of the features

used. For nonseasonal time-series, features that only apply to seasonal time-series are set to zero. Unlike the ES-RNN, the domain-specific features supplied with the M4 data set are not utilised in the FFORMA.

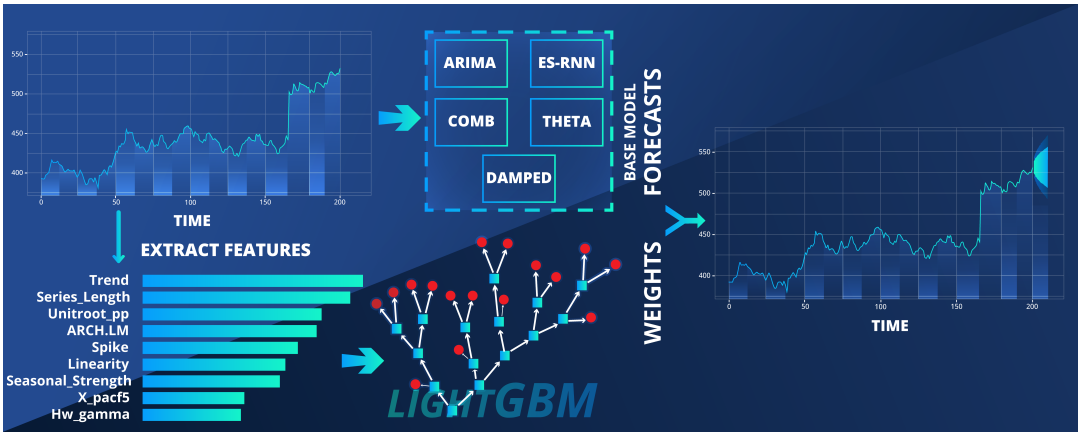


FIGURE 3.1: The proposed FFORMA forecasting pipeline.

3.3.3 The FFORMS

The Feature-based FORecast Model Selection (FFORMS) framework learns to select a single model from a pool of forecasting models according to their varying performance observed over some meta-data feature space [60].

A random forest ensemble learner [61] is adopted by the original FFORMS implementation to classify a single model as the most relevant for some time-series features extracted from the reference series. This study instead replaced the weighted-averaging of the FFORMA with the selection of the model with the highest allocated weighting. This change is highlighted in algorithm 1 (see Appendix A).

3.3.4 Feature-weighted Neural Stacking (NN-STACK)

Recently, Cawood and van Zyl [25] proposed model stacking to do forecasting of nonseasonal time-series. Their implementation performs regression over a feature space consisting of both the ensemble's model forecasts and

a set of statistics extracted from the time-series. This study builds on their work by experimenting with more meta-features and a more extensive data set of different domains and seasonality.

Meta-learning Model

The forecasts of a single timestep are combined using regression over both the ensemble's base model forecasts and the meta-features extracted from the input series. The MLP model performs regression over inputs of the model forecasts and the extracted meta-features, with the target variables as the predicted series' actual values.

A neural network of a basic architecture is adopted, and each layer is transformed using the ReLU activation function. The model is fit using mini-batches, and the Adam [62] stochastic gradient descent algorithm with a mean absolute error (MAE) loss function.

Meta-features

This study extends the previous implementation by including a more extensive set of meta-features for use by the FFORMA framework.

Feature selection is achieved using Spearman's rank correlation coefficient (ρ) [63], which measures the extent of the meta-features' correlation with the change in each model performance. This procedure helps reduce problems with overfitting, and only the most correlating features are included in the ensemble.

3.3.5 Feature-weighted Neural Averaging (NN-AVG)

This study proposes an additional MLP meta-learner that takes the time-series statistics as inputs and a one-hot encoded vector of the best performing model as the model's targets. The NN-AVG is a deep-learning approach to

the FFORMA's feature-weighted model averaging methodology. NN-AVG adopts a SoftMax activation function in the MLP's output layer to normalise the network's output to a probability distribution, i.e., the probability of each base model's adequacy to model a time series.

Meta-learning Model

The neural network is of a deep architecture and adopts a ReLU activation function at each layer except for the SoftMax activated output layer. The model is fit using mini-batches and the Adam stochastic gradient descent algorithm with a categorical cross-entropy loss function.

Forecasts Combination

The predictions of the forecasting models are combined, similar to the FFORMA's combination technique. The forecasts from the base models are summed after they are weighted according to the meta-learner's estimated probability distribution.

Meta-features

The same meta-features and feature selection procedure are used as per the NN-STACK approach.

3.4 Overview of Neural Basis Expansion Analysis (N-BEATS)

Oreshkin *et al.* [17] reported a 3% accuracy improvement over the forecasts of the ES-RNN. This study treats the N-BEATS method as a state of the art benchmark to compare the performance of the ensembles. Therefore, the

N-BEATS method is excluded from the ensembles' pool of base models. N-BEATS is a pure deep learning methodology that does not rely on feature engineering or input-scaling, and the rest of this section gives an overview of Oreshkin *et al.*'s [17] N-BEATS methodology.

3.4.1 Building Blocks

Oreshkin *et al.* [17] describe the N-BEATS core element as building blocks that accepts an input x_t and produces two outputs: $\hat{y}_{t+1}$, the forward forecast; and $\hat{x}_t$, the backcast (best estimate of x_t) that helps downstream blocks by removing noise from their inputs. The building blocks internally consists of two components. The first component is a linear connected network that generates the forward θ^f and the backward θ^b expansion coefficients. The second component consists of the backward g^b and the forward g^f basis layers that receive the expansion coefficients and project them internally on a set of basis functions to produce the block's $\hat{y}_{t+1}$ and $\hat{x}_t$ outputs.

The first component of the n-th block is described by the following set of equations:

$$h_{n,1} = \text{FC}_{n,1}(\text{h}_n) \tag{3.10}$$

$$h_{n,2} = \text{FC}_{n,2}(\text{h}_n, 1) \tag{3.11}$$

$$h_{n,3} = \text{FC}_{n,3}(\text{h}_n, 2) \tag{3.12}$$

$$h_{n,4} = \text{FC}_{n,4}(\text{h}_n, 3) \tag{3.13}$$

$$\theta^b_n = \text{LINEAR}^b_n(h_n, 4) \tag{3.14}$$

$$\theta^f_n = \text{LINEAR}^f_n(h_n, 4) \tag{3.15}$$

where FC denotes a fully connected layer with the nonlinear ReLU activation function and LINEAR is a linear projection layer. This architecture allows us to find the expansion coefficients for the forward and backcasts.

The second component of the n-th block uses the expansion coefficients to produce outputs with the basis layers using the following two equations:

$$\begin{aligned} \hat{y}_n &= g_n^f(\theta_n^f) \\ \hat{\mathrm{x}}_n &= g_n^b(\theta_n^b) \end{aligned} \tag{3.16}$$

where g_n^f and g_n^b are functional basis function layers. Different configurations are proposed for the functional basis layers g_n^f and g_n^b that produced better results for the different subsets of the M4 competition's data set. Oreshkin *et al.*'s [17] interpretation of these layers follows.

The Generic Architecture

This architecture is not dependent on knowledge regarding the time-series, and the building blocks instead learn the predictive decomposition of the block's forecast in the basis vector learned by the network. Layers g_n^f and g_n^b linearly projects the previous layer's output and the outputs of the n-th block are described as follows:

$$\begin{aligned} \hat{y}_n &= \mathrm{V}_n^f \, \theta_n^f + \mathrm{b}_n^f \\ \hat{\mathrm{x}}_n &= \mathrm{V}_n^b \, \theta_n^b + \mathrm{b}_n^b \end{aligned} \tag{3.17}$$

where V_n^f and V_n^f denote the forecast and backcast basis vectors and b is the bias.

Architecture to Model Trend

This layer mimics the trend behaviour of the time-series where the signal might be expressed using a monotonic function with some decay over time. This is achieved by constraining g_n^f and g_n^b to be a polynomial of degree p, a function varying slowly over the forecast horizon:

$$\hat{y}_n = \sum_{i=0}^{p} \theta_{n,i^t} \tag{3.18}$$

Time vector t is defined to have the layer forecast H steps ahead.

Architecture to Model Seasonality

To model time-series with seasonality, this layer mimics recurring fluctuation by imposing a constraint on g_n^f and g_n^b to have their outputs produce periodic changes depending on the class of seasonality (i.e., $y_t = y_t - \Delta$, where Δ is the period of the seasonality.)

The layer acts as a periodic function to produce outputs as a Fourier series using the following notation:

$$\hat{y}_n = \sum_{i=0}^{[H/2-1]} \theta_{n,i}^f \cos 2\pi i t + \theta_{n,i+[H/2]}^f \sin 2\pi i t \tag{3.19}$$

3.4.2 Doubly Residual Stacking

Oreshkin *et al.*'s [17] methodology extends the deep residual stacking [64] topology introduced by Huang *et al.* [36] in the DenseNet architecture. The novel architecture has two residual branches for the forecast and backcast predictions of each layer that might be found using the following equations:

$$\begin{aligned} \mathrm{x}_n &= \mathrm{x}_{n-1} - \hat{\mathrm{x}}_{n-1} \\ \hat{y} &= \sum_n \hat{y}_n \end{aligned} \tag{3.20}$$

The final forecast $\hat{y}$ from the doubly residual stacking is thus the sum of all partial forecasts from the blocks within a stack. Important to note here is that the time-series might be modelled by trend and seasonality decomposition by stacking blocks that model the trend and seasonality components.

3.4.3 Ensembling

A bagging procedure [65] is performed that includes 180 N-BEATS models trained with different random initialisations. The median is used as the ensemble aggregation function.

3.5 Details of Experimentation

3.5.1 The Evaluation Protocol

All ensembles are evaluated using ten-fold cross-validation, where each method is reinitialised ten times and evaluated for a 10% holdout set. This validation process is repeated for five runs, and the scores are averaged over all fifty validation sets to rule out chance from random initialisations.

A pseudorandom number generator (PRNG) [66] is configured to produce the indices for splitting the data into the training and validation sets. The seed numbers used are one, two, three, four and five for each run, respectively.

3.5.2 Research Instruments

Hardware and Software

All algorithms were implemented in Python and run on a 2.60GHz GHz Intel Core i7 PC with 16 16GB RAM and 1,365 MHz Nvidia RTX 2060 GPU with 6GB GDDR6 memory. The repository of this experiment's source code is available on GitHub [3].

[3] https://github.com/Pieter-Cawood/FFORMA-ESRNN

Data

The M4 data set is accessible on the Makridakis Open Forecasting Center's website [4] and consists of six subsets for each of the six frequencies to forecast. Each contains univariate time-series provided as numeric vectors without the conventional time stamp dependent variable. In addition, a data set is provided that contains features describing the domain and forecasting horizon for each of the reference sets.

For the weak-learner forecasts, the base models' point forecasts are reused from the M4 competition's submissions repository [5]. The meta-features are extracted from only the M4 competition's data training set for the ensembles that utilise them. For each cross-validation run, a different 90% splitting of the M4's test set is used as the ensembles' training target series and the other 10% as the holdout test set.

Third-party Libraries

The experiments use the Python implementation of the FFORMA [6] by Federico Garza and others that implement Microsoft's LightGBM method as the meta-learner. LightGBM has been shown to produce state of the art performance against other boosting methods [67]. The deep learning meta-learners (NN-STACK and NN-AVG) were implemented using Tensorflow's Keras module [7].

The *tsfeatures* [8] Python library was used to extract the same time-series features as per the original FFORMA implementation. Table B.1 (see Appendix B) provides a brief description of these features.

[4] https://mofc.unic.ac.cy/the-dataset
[5] https://github.com/Mcompetitions/M4-methods/tree/master/Point%20Forecasts
[6] https://github.com/FedericoGarza/fforma
[7] https://www.tensorflow.org/api_docs/python/tf/keras
[8] https://pypi.org/project/tsfeatures

The tuning of the gradient boosting hyperparameters was achieved using Bayesian optimisation with the Gaussian process method from the *Scikit-optimize* Python library [9] with minimisation configured over the OWA error.

Preprocessing of the Ensembles' Input Data

Only the stacking ensemble learned from the point forecasts of the base models directly. The study experimented with some preprocessing of the generated forecast data; however, the model's predictions were more accurate when fit with the raw forecast values.

Except for the simple model averaging, all ensembles utilise time-series statistics extracted from the reference series. No additional processing was required on this data since it was already statistical measurements.

The Ensembles' Architectures, Hyperparameters and Training

The ensembles' architectures and hyperparameters were found using the training set of the first fold of the first cross-validation run. The two ensembles that use neural networks (NN-STACK and NN-AVG) were tuned using backtesting, and the gradient boosting methods (FFORMA and FFORMS) were tuned using Bayesian optimisation. The parameter search space for the Bayesian optimisation was limited based on some initial results, and the Gaussian process method was configured to estimate the parameters that minimise the OWA error over 300 runs of parameter observations.

The **FFORMA** and **FFORMS** architecture and hyperparameters were identical, and they were automatically tuned using Bayesian optimisation over 300 runs of a limited parameter search space. Table 3.2 presents the hyperparameter settings used to train the FFORMA ensembles for the M4 competition's data set. Subset names H, W, D, M, Y and Q correspond to Hourly, Weekly, Daily, Monthly, Yearly and Quarterly data subsets.

[9] https://scikit-optimize.github.io

For the model's training, an early stopping procedure was implemented that monitors the loss of the validation set that is split as a quarter of the training data set. The early stopping patience was set to ten.

TABLE 3.2: The FFORMA's hyperparameters.

Hyperparameter	H	W	D	M	Y	Q
n-estimators	2000	2000	2000	1200	1200	2000
min data in leaf	63	50	200	100	100	50
number of leaves	135	19	94	110	110	94
eta	0.61	0.46	0.90	0.20	0.10	0.75
max depth	61	17	9	28	28	43
subsample	0.49	0.49	0.52	0.50	0.50	0.81
colsample bytree	0.90	0.90	0.49	0.50	0.50	0.49
early stop rounds	10	10	10	10	10	10

The **NN-STACK's** architectures and hyperparameters were determined using backtesting on the training data set of the first fold of the first run of cross-validation. The MLP architectures for the Hourly and Weekly subsets were deep with eleven hidden layers with neuron numbers of 100, 100, 100, 100, 100, 50, 50, 50, 50, 20 and 20 sequentially. A more shallow network was used to model the Daily, Monthly, Yearly and Quarterly subsets with three hidden layers with ten neurons each.

Early stopping was configured to terminate the training loop after a configured patience interval. Table 3.3 presents the hyperparameter settings used to train the NN-STACK ensembles for the M4 competition's data set. Subset names H, W, D, M, Y and Q correspond to Hourly, Weekly, Daily, Monthly, Yearly and Quarterly data subsets.

Similar to the stacking method, the **NN-AVG's** architecture and hyperparameters were determined using backtesting on the training data set of the first fold of the first run of cross-validation. The MLP architecture was deep, with eleven hidden layers of neurons in numbers of 100, 100, 100, 100, 100, 50, 50, 50, 50, 20 and 20 sequentially.

TABLE 3.3: The NN-STACK hyperparameters.

Hyperparameter	H	W	D	M	Y	Q
hidden layers	11	11	11	3	3	3
learning rate	0.0001	0.0001	0.0003	0.0003	0.0003	0.0003
batch size	1200	225	225	225	225	225
epochs	600	300	300	300	300	300
early stop rounds	15	15	15	15	15	15

Early stopping was configured to terminate the training loop after a configured patience interval. Table 3.4 presents the hyperparameter settings used to train the NN-AVG ensembles for the M4 competition's data set. Subset names H, W, D, M, Y and Q correspond to hourly, weekly, daily, monthly, yearly and quarterly data subsets.

TABLE 3.4: The NN-AVG hyperparameters.

Hyperparameter	H	W	D	M	Y	Q
hidden layers	11	11	11	11	11	11
learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
batch size	1200	225	52	52	52	52
epochs	600	300	300	300	300	300
early stop rounds	15	3	3	3	3	3

3.5.3 Results Data and Analysis

The forecasts of this study are across different horizons as per the competition requirements. The sMAPE, MASE and OWA errors are computed for the test set, which is split using ten-Fold cross-validation. The average OWA error is the critical instrument for analysing the results.

The forecasting methods' OWA error distributions are also visualised using violin plots to help analyse their performance. Hintze and Nelson [68] proposed violin plot as an extension to the traditional box plot that integrates a smoothed histogram from local density estimates to display the data in a single helpful plot for analysis.

3.6 Limitations

This study is limited to experimentation with a handful of ensemble techniques that showed promise to deliver state of the art forecasting performance.

The performance of the ensembles is not analysed in terms of their speed as we are only concerned regarding their accuracy as per the M4 competition requirements.

This study only reports the model architectures and hyperparameters of the ensembles implemented in this report, as the forecasts of the base models are retrieved from a secondary source, and their parameters are irrelevant.

3.7 Ethical Considerations

This research does not require ethical clearance as it does not contain personal data and is freely available to the public.

Chapter 4

Results and Discussion

This section reports the forecasting results of each method for the six data subsets of the M4 forecasting competition. The results are reported for all ensembles over fifty runs of independent data sets using the evaluation protocol discussed in section 3.5.1.

4.1 Detailed Observations

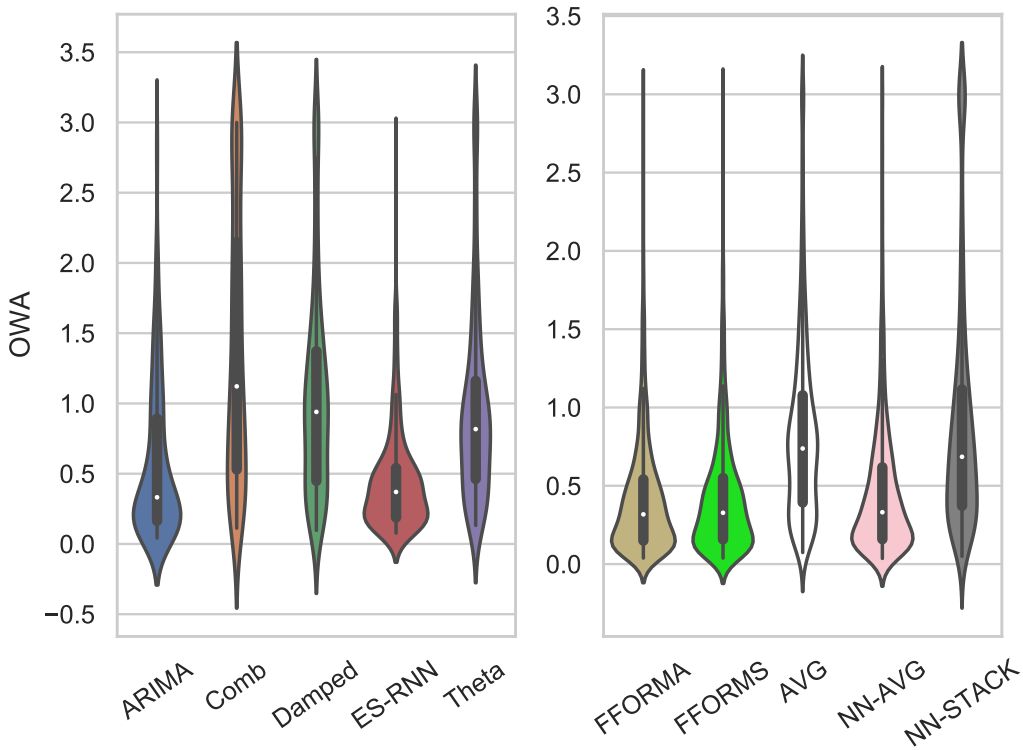
Table 4.1 presents key empirical results for the quantitative analysis. The table presents the average OWA performance for each forecasting method for each of the six data subsets of the M4 forecasting competition. Figures 4.1 - 4.6 depict the model OWA distributions and visualises the data quartiles and extreme values. The visualisations of the error distributions are used to analyse the consistency of each forecast method's accuracy. The violin plots' upper extreme values were clipped to 3.0, a high OWA measurement representing model failure.

Table 4.1's column names H, W, D, M, Y and Q correspond to the Hourly, Weekly, Daily, Monthly, Yearly and Quarterly data subsets. Column "H,W & D" presents the average over the three smaller subsets, and the average over all six subsets is given in the last column. The number of time series of each data subset is provided in parenthesis, the best results are given in bold, and the top two are highlighted in grey.

TABLE 4.1: Average OWA performance over fifty test-folds of the M4 data set. N-BEATS† reproduced here for comparison.

	H (0.4K)	W (0.4K)	D (4.2K)	H,W&D	M (48K)	Y (23K)	Q (24K)	Average
Base models								
ARIMA	0.577	0.925	1.047	0.850	0.903	0.892	0.898	0.874
Comb	1.556	0.944	0.981	1.160	0.920	0.867	0.890	1.026
Damped	0.936	0.992	0.999	0.976	0.924	0.890	0.893	0.939
ES-RNN	0.440	0.864	1.046	0.783	0.836	0.778	0.847	0.802
Theta	1.006	0.965	0.998	0.990	0.907	0.872	0.917	0.944
Ensembles								
FFORMA	**0.415**	**0.725**	0.983	**0.708**	**0.800**	**0.732**	0.816	**0.745**
FFORMS	0.427	0.740	0.984	0.717	0.810	0.745	0.826	0.755
AVG	0.847	0.860	0.985	0.897	0.863	0.804	0.856	0.869
NN-AVG	0.453	0.740	0.986	0.726	0.825	0.759	0.839	0.767
NN-STACK	1.427	0.810	**0.927**	1.055	0.833	0.771	0.838	0.934
State of the Art								
N-BEATS† [17]	-	-	-	0.822	0.833	0.819	**0.758**	0.795

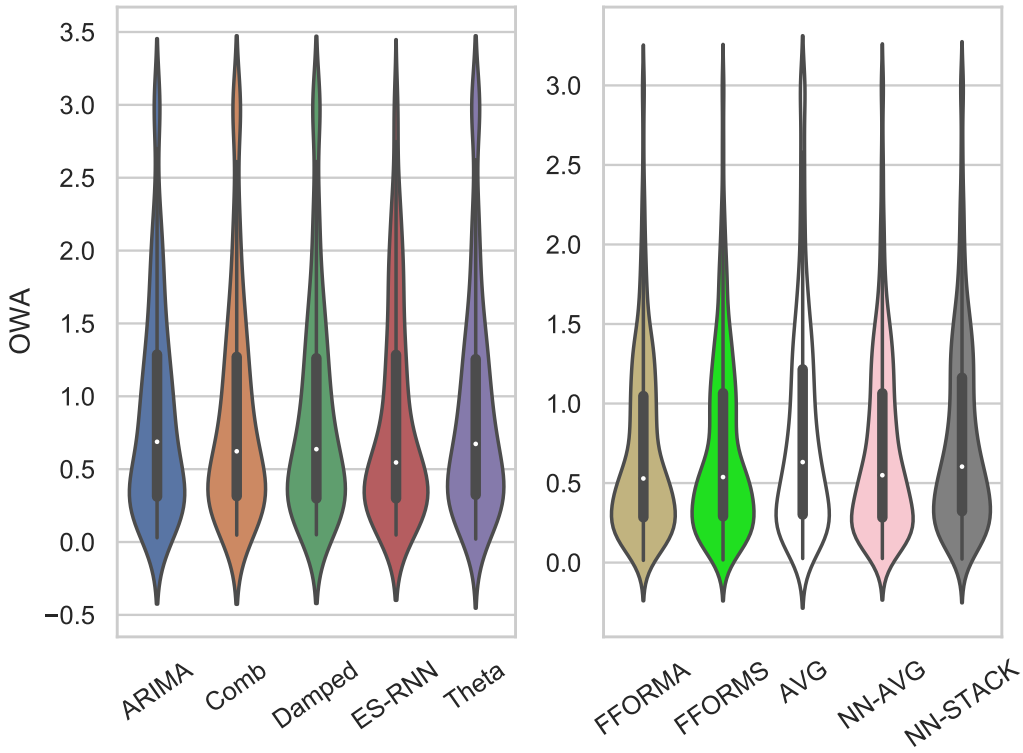
FIGURE 4.1: Hourly subset OWA distributions



4.1.1 Hourly Subset

For the Hourly subset, the ES-RNN produced the only stable forecasts amongst the pool of base models (see Figure 4.1.) Subsequently, the NN-STACK failed

FIGURE 4.2: Weekly subset OWA distributions



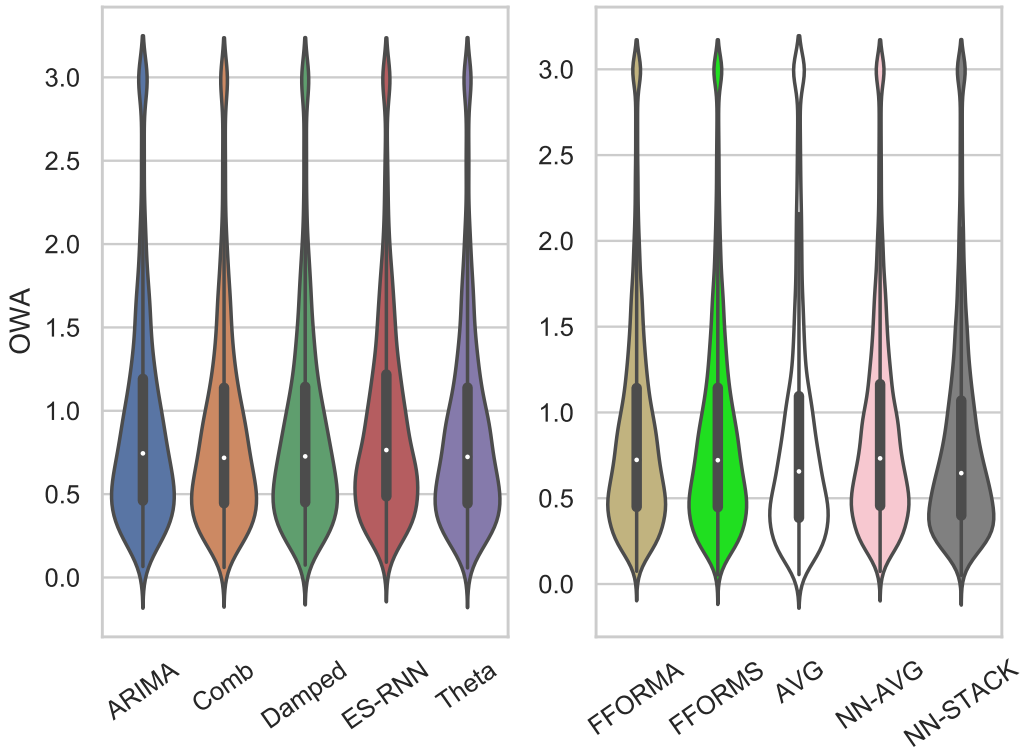
as an ensembling method and produced the experiment's worst average ensemble result. This failure was partly due to the small data set of only 414 series and the largest forecasting horizon requirement of 48 points.

The FFORMA outperformed all other forecasting methods and produced results at least twice as good as the AVG benchmark. It was noted here that the ES-RNN's average OWA error might slightly be improved ($0.440 - 0.415 = 0.025$) by utilising the hybrid model with the FFORMA ensemble technique.

4.1.2 Weekly Subset

Considering the OWA distributions depicted in Figure 4.2, the ES-RNN produced the lowest average OWA error amongst the other base models, with

FIGURE 4.3: Daily subset OWA distributions



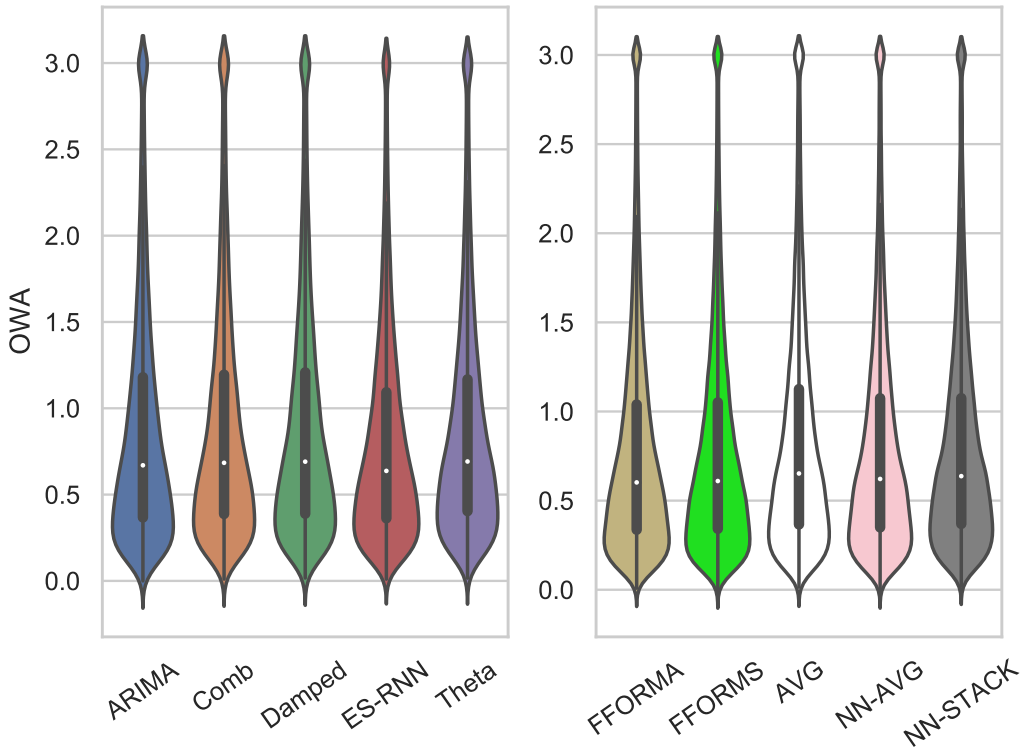
considerably fewer forecasts reaching the extreme value of 3.0. This distinctive performance of at least one base model allowed all ensemble methods to outperform all base models (see Table 4.1.)

The FFORMA ensemble improved the ES-RNN's OWA error by a large margin ($0.864 - 0.725 = 0.139$). The FFORMA performed the best of all methods, followed by the NN-AVG and FFORMA as the second-best, whereas the NN-AVG was the best performing neural network based ensemble approach.

4.1.3 Daily Subset

From the distributions of base model errors (see Figure 4.3), it was noted that the pool of base models performed exceptionally similar. Moreover, this was the only data set where the ES-RNN failed to outperform other base models. Subsequently, the gradient boosting methods failed to gain the advantage of

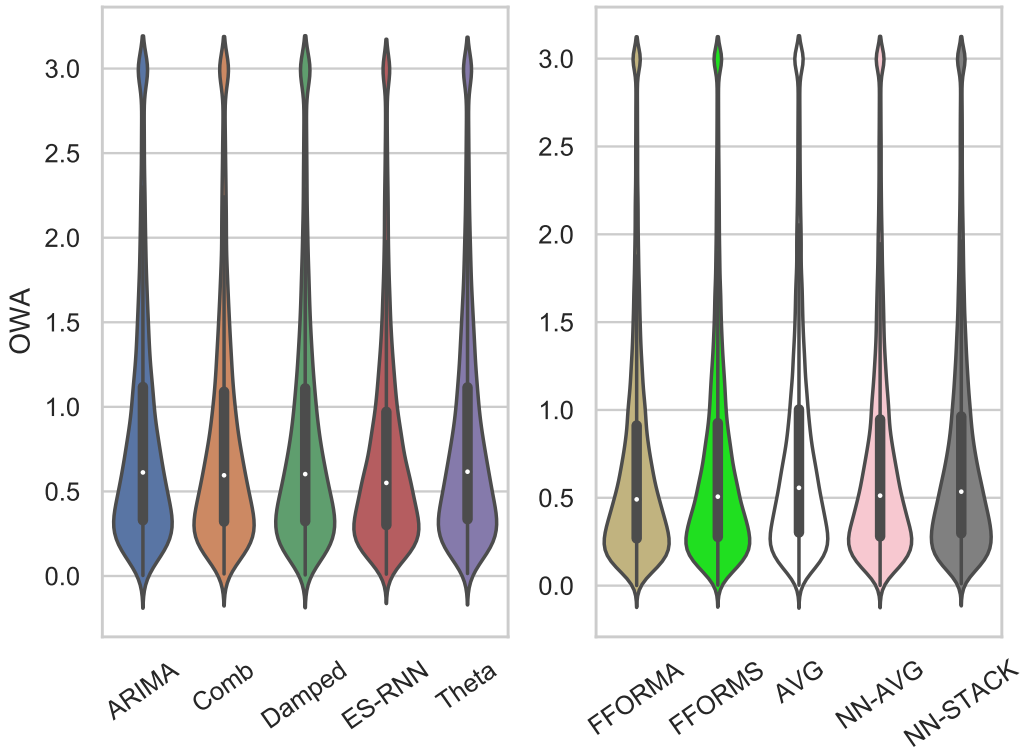
FIGURE 4.4: Monthly subset OWA distributions



the ES-RNN's forecasts as per the other subsets, and the NN-STACK method was the only successful ensemble and best approach.

The Daily subset contains the most imbalanced time-series from the different domains. Subsequently, the gradient boosting methods failed to perform since they do not utilise information regarding the time-series domains. This was also the only experiment where significantly similar performance was observed amongst the base models. The results suggest that the NN-STACK method is a more suitable ensemble for situations where it is difficult for the ensemble to learn the weightings of superiority amongst the pool of base models.

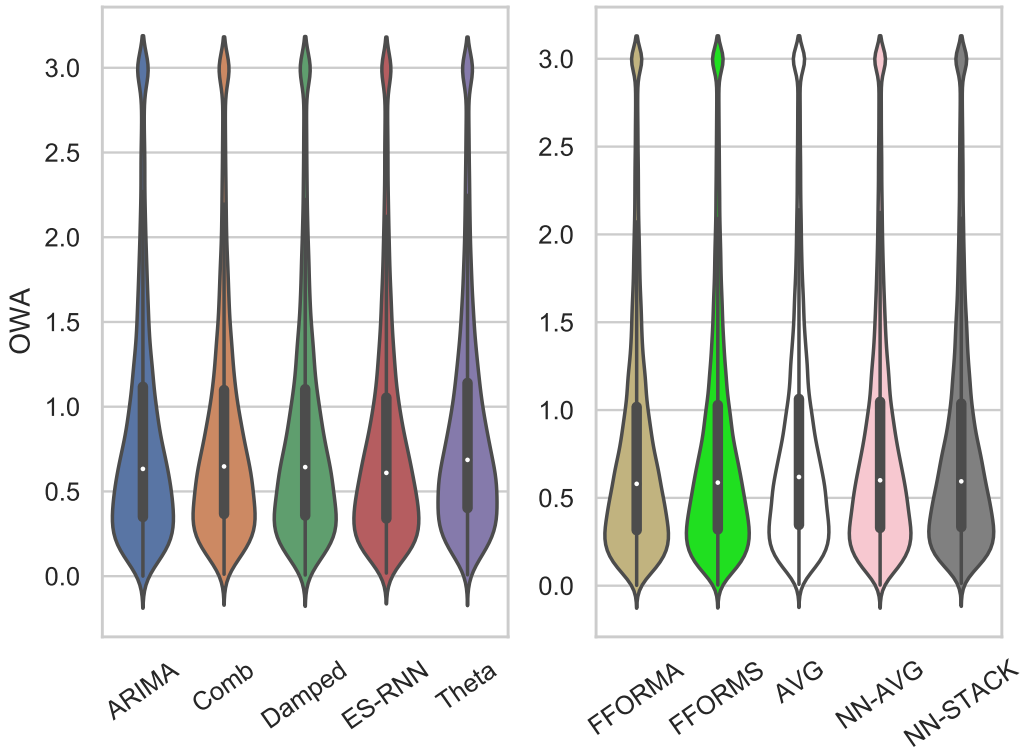
FIGURE 4.5: Yearly subset OWA distributions



4.1.4 Monthly Subset

The Monthly subset consists of 48,000 time-series, and it was the largest- and most domain-balanced data set of all the experiments. Figure 4.4 depicts the distribution of the OWA errors, and the ES-RNN performed best amongst the base models, with a notably smaller number of model failures. These conditions are ideal for analysing the general performance of the ensemble methods, and the results show similar performance in average OWA errors and only a slight ($0.836 - 0.800 = 0.036$) improvement when the FFORMA learns from the predictions of the ES-RNN. Nevertheless, considering the average OWA error, the FFORMA remained the superior method and outperformed all other approaches.

FIGURE 4.6: Quarterly subset OWA distributions



4.1.5 Yearly Subset

The Yearly subset was one of the experiments' larger and nonseasonal data sets. It was noted that all ensembles, except for the AVG benchmark, slightly outperformed the ES-RNN. Similar OWA distributions were observed between the gradient boosting and neural network ensembles (see Figure 4.5.) However, the FFORMA showed the greatest improvement ($0.778 - 0.732 = 0.046$), and it is the preferred ensemble for time-series with a Yearly seasonality.

4.1.6 Quarterly Subset

The Quarterly subset is similar to the Yearly one in size, balance and forecast horizon. The only exception is that the Quarterly time-series contain a higher seasonality.

Considering the ensembles' OWA distributions visualised in Figure 4.6, similar ensembling performance was observed to that of the ensembling performance on the Yearly subset. Likewise, no base model was distinguishable in terms of the OWA distributions. However, the larger data set allowed the ensemble methods to avoid the case of the smaller Daily data set (with similar performing base models), and the FFORMA ensemble produced a slight improvement ($0.847 - 0.816 = 0.031$) over the average accuracy of the ES-RNN.

The N-BEATS approach showed the best results for the Quarterly subset, and it was able to outperform the best ensemble for this subset, FFORMA, by a small margin ($0.816 - 0.758 = 0.058$.)

4.1.7 Overall Results

The N-BEATS outperformed all base models for the Quarterly subset. Consequently, it was the only case where the N-BEATS outperformed all other ensembles. Therefore, the utility of N-BEATS might have notably boosted the ensembles' accuracy for the Quarterly subset.

The AVG method was added as a benchmark for the other ensembles to analyse their performance relative to something intuitive. The average OWA gap between the AVG and other ensemble methods (e.g., between the FFORMA: $0.869 - 0.745 = 0.124$) indicate that ML is a robust solution to late data fusion.

Although all the ensembles beat the ES-RNN's average error considering the OWA distributions (Figures 4.4 - 4.6) of the three larger subsets (Monthly, Yearly and Quarterly), there is still generally a deal of uncertainty. Nevertheless, the ensembles were still practical to lower the upper quantiles of the OWA distributions by a notable margin. A larger improvement was observed in the OWA distributions of the three smaller subsets (see Figures 4.1 -

4.3), where the weak learners showed more distinctive performance (considering their average OWA errors.) This distinctive ensembling performance provides evidence that i) ensemble learning's performance is dependent on the distinctive performances of its weak learners and ii) ensemble learning's performance is dependent on the diversity of the data (i.e., the similarity of the time-series from the different domains.)

The gradient boosting ensembles generally outperformed those of neural networks. The average OWA gap between the top two methods of each (i.e., the FFORMA and NN-AVG: 0.767 - 0.745 = 0.022) is enough to confirm the superiority of gradient boosting as an ensemble method for time-series forecasting. Furthermore, both gradient boosting ensembles outperformed all other ensemble methods on all subsets except for the Daily subset. Table 4.1 statistically shows that for all data sets, it is always possible to improve a standalone time-series forecast model by adding it to an ensemble with other statistical forecast models. Lastly, since the FFORMA utilises the ES-RNN as a base model, the FFORMA could outperform the standalone ES-RNN (the M4 winner) for all six subsets of data.

Chapter 5

Conclusion

The primary objective of this study was to compare ensemble methods against the state of the art forecasting methods, the ES-RNN and N-BEATS, to determine whether the study might i) improve the accuracy of the ES-RNN and ii) single out any method as the state of the art ensembling technique for future research efforts.

After a review of time-series forecasting research, a gap was noted in research that determines the boost of accuracy one might achieve from utilising a state of the art forecasting method in an ensemble of models. Moreover, research was needed to compare the performance of different types of ensembles using the same forecasts from a pool of forecasting models. Consequently, this study's research questions were aligned with answering the extent to which the forecast accuracy of a state of the art forecasting method like the ES-RNN might be improved from ensemble learning.

The experiment used a ten-fold cross-validation procedure with the M4 forecasting competition's data set. The experiment included four statistical base models, namely ARIMA, Comb, Damped and Theta. And an advanced machine learning and statistical hybrid model, the ES-RNN. The base models were used with different ensemble strategies of model averaging, stacking, and selection. The ensembles of the experiment include two gradient

boosting ones, namely the FFORMA and FFORMS; two neural network approaches, namely NN-STACK and NN-AVG; and a naive arithmetic average as an ensemble benchmark. The study additionally compares the results of another state of the art benchmark, namely the N-BEATS, which was excluded from the pool of base models for the ensembles.

In respect of the obtained results, this study shows that the FFORMA is a state of the art ensemble technique that might be used to boost the performance of powerful methods like the ES-RNN and N-BEATS. However, considering the case of the Daily subset, the NN-STACK approach might be a more suitable tool when the pool of forecasting models have similar performance. None of the ensembles was able to prevent model failure (where the measured OWA was greater than 3.0). Further, this study shows that weighted model averaging is a superior ensemble approach as both the FFORMA and NN-AVG outperformed versions of their same architecture that do model stacking or selection.

The stacking approach of this study performs regression over single points of the base model forecasts. NN-STACK, therefore, ignores the temporal element of the data, and the meta-learner instead learns a generalised combination function. Consequently, it is recommended for further research to consider investigating the performance of a stacking ensemble that is conscious of the data's temporality. Furthermore, considering the equal-weighted averaging adopted by the top submissions of the M5 competition, further research is needed to assess the performance of ensembling techniques with hierarchical time-series data.

Appendix A

The FFORMA algorithm

Algorithm 1: FFORMA's forecast combination [15].

OFFLINE PHASE: TRAINING

Inputs

$\{x_1, x_2, \ldots, x_N\}$: N observed time series from the reference set.
$\{f_1, f_2, \ldots, f_F\}$: a set of F functions to calculate the meta-features.
$\{m_1, m_2, \ldots, m_M\}$: a set of M forecasting models.

Output

Meta-learner model

Prepare the metadata

for $n \leftarrow 1$ ***to*** N **do**

1. Split x_n to a training and test series using time-splitting.
2. Calculate the meta-features $f_n \in F$ over the training period.
3. Fit each base forecasting method $m \in M$ and generate forecasts.
4. Compute the forecast losses L_{nm} over the test period.

end

Train the meta-learner

Train the meta-learner model using the meta-data and forecasting errors of the base models, by minimising:

$$\arg\min_{w} \sum_{n=1}^{N} \sum_{m=1}^{M} w(f_n)_m L_{nm}$$

ONLINE PHASE: FORECASTING

Inputs

The meta-learner model from the offline phase.
$\{x_1, x_2, \ldots, x_N\}$: N observed time series from the validation set.
$\{m_1, m_2, \ldots, m_M\}$: the set of M forecasting models.

Output

$\{y_1, y_2, \ldots, y_N\}$: Forecast for each series in the test set.

for $n \leftarrow 1$ ***to*** N **do**

1. Calculate the meta-features $f_n \in F$.
2. Use the meta-learner to produce $w(f_n)$, an M-vector of weights.
3. Generate the forecasts for each of the M forecasting methods.
4. Combine each of the forecasts using w to produce the final forecast.

3-4. FFORMS: Select from M the model with the highest allocated w to produce the final forecast.

end

*FFORMS modification that replace the FFORMA forecast steps 3 & 4.

Appendix B

The FFORMA meta-features

TABLE B.1: Features used in the FFORMA framework [15].

	Feature	Description	Nonseasonal	Seasonal
1	T	length of time series	Yes	Yes
2	trend	strength of trend	Yes	Yes
3	seasonality	strength of seasonality	-	Yes
4	linearity	linearity	Yes	Yes
5	curvature	curvature	Yes	Yes
6	spikiness	spikiness	Yes	Yes
7	e_acf1	first ACF value of remainder series	Yes	Yes
8	e_acf10	sum of squares of first 10 ACF values of remainder series	Yes	Yes
9	stability	stability	Yes	Yes
10	lumpiness	lumpiness	Yes	Yes
11	entropy spectral	entropy	Yes	Yes
12	hurst	Hurst exponent	Yes	Yes
13	nonlinearity	nonlinearity	Yes	Yes
14	alpha	$ETS(A,A,N)\hat{\alpha}$	Yes	Yes
15	beta	$ETS(A,A,N)\hat{\beta}$	Yes	Yes
16	hwalpha	$ETS(A,A,A)\hat{\alpha}$	-	Yes
17	hwbeta	$ETS(A,A,A)\hat{\beta}$	-	Yes
18	hwgamma	$ETS(A,A,A)\hat{\gamma}$	-	Yes
19	ur_pp	test statistic based on Phillips–Perron test	Yes	Yes
20	ur_kpss	test statistic based on KPSS test	Yes	Yes
21	y_acf1	first ACF value of the original series	Yes	Yes
22	diff1y_acf1	first ACF value of the differenced series	Yes	Yes
23	diff2y_acf1	first ACF value of the twice-differenced series	Yes	Yes
24	y_acf10	sum of squares of first 10 ACF values of original series	Yes	Yes
25	diff1y_acf10	sum of squares of first 10 ACF values of differenced series	Yes	Yes
26	diff2y_acf10	sum of squares of first 10 ACF values of twice-differenced series	Yes	Yes
27	seas_acf1	autocorrelation coefficient at first seasonal lag	-	Yes
28	sediff_acf1	first ACF value of seasonally differenced series	-	Yes
29	y_pacf5	sum of squares of first 5 PACF values of original series	Yes	Yes
30	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	Yes	Yes
31	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	Yes	Yes
32	seas_pacf	partial autocorrelation coefficient at first seasonal lag	Yes	Yes
33	crossing_point	number of times the time series crosses the median	Yes	Yes
34	flat_spots	number of flat spots from 10 equidistant points	Yes	Yes
35	nperiods	number of seasonal periods in the series	-	Yes
36	seasonal_period	length of seasonal period	-	Yes
37	peak	strength of peak	Yes	Yes
38	trough	strength of trough	Yes	Yes
39	ARCH.LM	ARCH LM statistic	Yes	Yes
40	arch_acf	sum of squares of the first 12 autocorrelations of z2	Yes	Yes
41	garch_acf	sum of squares of the first 12 autocorrelations of r2	Yes	Yes
42	arch_r2	R2 value of an AR model applied to z2	Yes	Yes
43	garch_r2	R2 value of an AR model applied to r2	Yes	Yes

Bibliography

[1] S. Makridakis, R. J. Hyndman, and F. Petropoulos, "Forecasting in social settings: The state of the art," *International Journal of Forecasting*, vol. 36, no. 1, pp. 15–28, 2020.

[2] J. Burton, "Robert fitzroy and the early history of the meteorological office," *The British Journal for the History of Science*, pp. 147–176, 1986.

[3] G. Udny Yule, "On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers," *Philosophical Transactions of the Royal Society of London Series A*, vol. 226, pp. 267–298, 1927.

[4] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[5] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[6] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m5 competition: Background, organization, and implementation," *International Journal of Forecasting*, 2021.

[7] N. G. Reich, L. C. Brooks, S. J. Fox, *et al.*, "A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the united states," *Proceedings of the National Academy of Sciences*, vol. 116, no. 8, pp. 3146–3154, 2019.

[8] C. J. McGowan, M. Biggerstaff, M. Johansson, *et al.*, "Collaborative efforts to forecast seasonal influenza in the united states, 2015–2016," *Scientific reports*, vol. 9, no. 1, pp. 1–13, 2019.

[9] M. A. Johansson, K. M. Apfeldorf, S. Dobson, *et al.*, "An open challenge to advance probabilistic forecasting for dengue epidemics," *Proceedings of the National Academy of Sciences*, vol. 116, no. 48, pp. 24 268–24 274, 2019.

[10] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *International Journal of Forecasting*, 2022.

[11] L. Rokach, "Ensemble-based classifiers," *Artificial intelligence review*, vol. 33, no. 1, pp. 1–39, 2010.

[12] Q. Wu, "A hybrid-forecasting model based on gaussian support vector machine and chaotic particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2388–2394, 2010.

[13] M. Khashei and M. Bijari, "A new class of hybrid models for time series forecasting," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4344–4357, 2012.

[14] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLoS ONE*, vol. 13, Mar. 2018.

[15] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "Fforma: Feature-based forecast model averaging," *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, 2020.

[16] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020, M4 Competition.

[17] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[18] G. R. Shinde, A. B. Kalamkar, P. N. Mahalle, N. Dey, J. Chaki, and A. E. Hassanien, "Forecasting models for coronavirus disease (covid-19): A survey of the state-of-the-art," *SN Computer Science*, vol. 1, no. 4, pp. 1–15, 2020.

[19] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[20] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[21] B Arinze, "Selecting appropriate forecasting models using rule induction," *Omega*, vol. 22, no. 6, pp. 647–658, 1994.

[22] A. E. Raftery, D. Madigan, and J. A. Hoeting, "Bayesian model averaging for linear regression models," *Journal of the American Statistical Association*, vol. 92, no. 437, pp. 179–191, 1997.

[23] B. Clarke, "Comparing bayes model averaging and stacking when model approximation error cannot be ignored," *Journal of Machine Learning Research*, vol. 4, no. Oct, pp. 683–712, 2003.

[24] M. H. D. M. Ribeiro and L. dos Santos Coelho, "Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series," *Applied Soft Computing*, vol. 86, p. 105 837, 2020.

[25] P. Cawood and T. L. van Zyl, "Feature-weighted stacking for nonseasonal time series forecasts: A case study of the covid-19 epidemic curves,"

in *2021 8th International Conference on Soft Computing Machine Intelligence (ISCMI)*, 2021, pp. 53–59.

[26] A. C. Lorena, A. I. Maciel, P. B. de Miranda, I. G. Costa, and R. B. Prudêncio, "Data complexity meta-features for regression problems," *Machine Learning*, vol. 107, no. 1, pp. 209–246, 2018.

[27] S. Barak, M. Nasiri, and M. Rostamzadeh, "Time series model selection with a meta-learning approach; evidence from a pool of forecasting algorithms," *arXiv preprint arXiv:1908.08489*, 2019.

[28] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[29] N. Liu, Q. Tang, J. Zhang, W. Fan, and J. Liu, "A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids," *Applied Energy*, vol. 129, pp. 336–345, 2014.

[30] J.-Z. Wang, Y. Wang, and P. Jiang, "The study and application of a novel hybrid forecasting model–a case study of wind speed forecasting in china," *Applied Energy*, vol. 143, pp. 472–488, 2015.

[31] Y. Qin, K. Li, Z. Liang, *et al.*, "Hybrid forecasting model based on long short term memory network and deep learning neural network for wind signal," *Applied energy*, vol. 236, pp. 262–272, 2019.

[32] A. Aksoy, N. Öztürk, and E. Sucky, "Demand forecasting for apparel manufacturers by using neuro-fuzzy techniques," *Journal of Modelling in Management*, 2014.

[33] W. Deng, G. Wang, and X. Zhang, "A novel hybrid water quality time series prediction method based on cloud model and fuzzy forecasting," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 39–49, 2015.

[34] R. Rahmani, R. Yusof, M. Seyedmahmoudian, and S. Mekhilef, "Hybrid technique of ant colony and particle swarm optimization for short term wind energy forecasting," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 123, pp. 163–170, 2013.

[35] S. Kumar, S. K. Pal, and R. Singh, "A novel hybrid model based on particle swarm optimisation and extreme learning machine for short-term temperature prediction using ambient sensors," *Sustainable Cities and Society*, vol. 49, p. 101 601, 2019.

[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[37] G. S. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques–part ii: Soft computing methods," *Expert systems with applications*, vol. 36, no. 3, pp. 5932–5941, 2009.

[38] J. E. Hanke and D. W. Wichern, "Business forecasting," 2005.

[39] J. Fattah, L. Ezzine, Z. Aman, H. El Moussami, and A. Lachhab, "Forecasting of demand using arima model," *International Journal of Engineering Business Management*, vol. 10, p. 1 847 979 018 808 673, 2018.

[40] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for r," *Journal of statistical software*, vol. 27, no. 1, pp. 1–22, 2008.

[41] V. Assimakopoulos and K. Nikolopoulos, "The theta model: A decomposition approach to forecasting," *International journal of forecasting*, vol. 16, no. 4, pp. 521–530, 2000.

[42] E. Ostertagová and O. Ostertag, "The simple exponential smoothing model," in *The 4th International Conference on Modelling of Mechanical*

and Mechatronic Systems, Technical University of Košice, Slovak Republic, Proceedings of conference, 2011, pp. 380–384.

[43] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International journal of forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[44] E. McKenzie and E. S. Gardner Jr, "Damped trend exponential smoothing: A modelling viewpoint," *International Journal of Forecasting*, vol. 26, no. 4, pp. 661–665, 2010.

[45] S. Makridakis and M. Hibon, "The m3-competition: Results, conclusions and implications," *International journal of forecasting*, vol. 16, no. 4, pp. 451–476, 2000.

[46] A. Nazim and A. Afthanorhan, "A comparison between single exponential smoothing (ses), double exponential smoothing (des), holt's (brown) and adaptive response rate exponential smoothing (arres) techniques in forecasting malaysia population," *Global Journal of Mathematical Analysis*, vol. 2, no. 4, pp. 276–280, 2014.

[47] P. S. Kalekar *et al.*, "Time series forecasting using holt-winters exponential smoothing," *Kanwal Rekhi school of information Technology*, vol. 4329008, no. 13, pp. 1–13, 2004.

[48] O. Fathi, "Time series forecasting using a hybrid arima and lstm model," *Velvet Consulting*, 2019.

[49] S. Chang, Y. Zhang, W. Han, *et al.*, "Dilated recurrent neural networks," *arXiv preprint arXiv:1710.02224*, 2017.

[50] F. Petropoulos, R. J. Hyndman, and C. Bergmeir, "Exploring the sources of uncertainty: Why does bagging for time series forecasting work?" *European Journal of Operational Research*, vol. 268, no. 2, pp. 545–554, 2018.

[51] F. Chan and L. L. Pauwels, "Some theoretical results on forecast combinations," *International Journal of Forecasting*, vol. 34, no. 1, pp. 64–74, 2018.

[52] E. S. Gardner Jr, "Exponential smoothing: The state of the art," *Journal of forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[53] R. J. Chen, P. Bloomfield, and J. S. Fu, "An evaluation of alternative forecasting methods to recreation visitation," *Journal of Leisure Research*, vol. 35, no. 4, pp. 441–454, 2003.

[54] S. Makridakis, "Accuracy measures: Theoretical and practical concerns," *International journal of forecasting*, vol. 9, no. 4, pp. 527–529, 1993.

[55] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International journal of forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[56] A. Redd, K. Khin, and A. Marini, "Fast es-rnn: A gpu implementation of the es-rnn algorithm," *arXiv preprint arXiv:1907.03329*, 2019.

[57] G. Claeskens, N. L. Hjort, *et al.*, "Model selection and model averaging," *Cambridge Books*, 2008.

[58] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[59] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *2015 IEEE international conference on data mining workshop (ICDMW)*, IEEE, 2015, pp. 1616–1619.

[60] T. S. Talagala, R. J. Hyndman, G. Athanasopoulos, *et al.*, "Meta-learning how to forecast time series," *Monash Econometrics and Business Statistics Working Papers*, vol. 6, p. 18, 2018.

[61] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[63] J. H. Zar, "Spearman rank correlation," *Encyclopedia of biostatistics*, vol. 7, 2005.

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[65] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[66] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on computing*, vol. 15, no. 2, pp. 364–383, 1986.

[67] E. Al Daoud, "Comparison between xgboost, lightgbm and catboost using a home credit dataset," *International Journal of Computer and Information Engineering*, vol. 13, no. 1, pp. 6–10, 2019.

[68] J. L. Hintze and R. D. Nelson, "Violin plots: A box plot-density trace synergism," *The American Statistician*, vol. 52, no. 2, pp. 181–184, 1998.