



PALGRAVE ADVANCES IN THE  
ECONOMICS OF INNOVATION AND  
TECHNOLOGY

# Forecasting with Artificial Intelligence

## Theory and Applications

*Edited by*

MOHSEN HAMOUDIA  
SPYROS MAKRIDAKIS  
EVANGELOS SPILIOOTIS

palgrave  
macmillan

Palgrave Advances in the Economics of Innovation  
and Technology

Series Editor

Albert N. Link, Department of Economics, University of North  
Carolina at Greensboro, Greensboro, NC, USA

The focus of this series is on scholarly inquiry into the economic foundations of technologies and the market and social consequences of subsequent innovations. While much has been written about technology and innovation policy, as well as about the macroeconomic impacts of technology on economic growth and development, there remains a gap in our understanding of the processes through which R&D funding leads to successful (and unsuccessful) technologies, how technologies enter the market place, and factors associated with the market success (or lack of success) of new technologies.

This series considers original research into these issues. The scope of such research includes in-depth case studies; cross-sectional and longitudinal empirical investigations using project, firm, industry, public agency, and national data; comparative studies across related technologies; diffusion studies of successful and unsuccessful innovations; and evaluation studies of the economic returns associated with public investments in the development of new technologies.

Mohsen Hamoudia · Spyros Makridakis ·  
Evangelos Spiliotis  
Editors

# Forecasting with Artificial Intelligence

Theory and Applications

palgrave  
macmillan

*Editors*

Mohsen Hamoudia  
France Telecom Group  
Orange Business Services  
Eragny, France

Spyros Makridakis  
Institute For the Future (IFF)  
University of Nicosia  
Engomi, Cyprus

Evangelos Spiliotis  
School of Electrical and Computer  
Engineering  
National Technical University  
of Athens  
Zografou, Greece

ISSN 2662-3862

ISSN 2662-3870 (electronic)

Palgrave Advances in the Economics of Innovation and Technology

ISBN 978-3-031-35878-4

ISBN 978-3-031-35879-1 (eBook)

<https://doi.org/10.1007/978-3-031-35879-1>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Palgrave Macmillan imprint is published by the registered company Springer Nature Switzerland AG

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To Our Families  
To the International Institute of Forecasters community*

## FOREWORD

As I walk along the beach and watch the incoming tide, the variety of waves is seemingly endless. Some are small and quickly overtaken by other stronger waves following on behind. Others are large enough to herald their arrival with cresting white tops before crashing onto the shore and splashing or even soaking the unwary.

The evolution of forecasting methodology exhibits similar surges interspersed with periods of calm or occasional retreats. However, unlike the ocean, the field of forecasting is able to continue its progress as earlier causal and extrapolative procedures are enhanced by machine learning techniques, which is the focus of this volume.

There has been some recent concern that Artificial Intelligence (AI) is rogue science. Although most of the book focuses on the subset of AI that is machine learning (ML), the authors clearly embrace AI in its broadest context, to the extent of a preface written by ChatGPT. (Hey, Chat, for your information you can't rhyme *harness* with *harness*, try *farness*.)

After a broad overview of AI related to forecasting, most chapters provide state-of-the-art discussions of the impact of ML upon forecasting activity relating to time series analysis, global forecasting models, large data analysis, combining forecasts, and model selection. Even newer topics such as concept drift and meta-learning sent me a-googling for definitions. The remaining chapters include case studies in economics and operations research; the finale is a chapter on Forecast Value Added (FVA), a down-to-earth method for determining whether your attempts

to add value by modifying your forecasting process are making things better (or worse!).

As should be evident from this brief summary, this ensemble of papers will prove invaluable to anyone seeking to understand recent developments in this rapidly growing area of research.

Of course, some of the old questions remain far from settled. Are we seeking explanation and understanding or is the focus purely on prediction? Can we interpret the results to the satisfaction of a skeptical consumer (or manager)? Last but not least, who decides what data to input to the forecasting process? GIGO (garbage in, garbage out) never goes away. These questions will continue to provide interesting challenges for researchers at the AI/Forecasting interface.

AI may lack a moral compass and a sense of humor, but this volume demonstrates its undoubted value in improving the forecasting process and should be on every forecaster's bookshelf.

May 2023

Keith Ord  
McDonough School of Business  
Georgetown University  
Washington, DC, USA

# PREFACE

This book explores the intersection of Artificial Intelligence (AI) and forecasting, providing an overview of the current capabilities and potential implications of the former for the theory and practice of forecasting. It contains 14 chapters that touch on various topics, such as the concept of AI, its impact on economic decision-making, traditional and machine learning-based forecasting methods, challenges in demand forecasting, global forecasting models, including key illustrations, state-of-the-art implementations, best practices and notable advances, meta-learning and feature-based forecasting, ensembling, deep learning, scalability in industrial and optimization applications, and forecasting performance evaluation.

The book delves into the challenges and opportunities of using AI in time series forecasting, discussing ways to improve forecasting accuracy, handle non-stationary data, and address data scalability and computational efficiency issues. It also explores the interpretability and explainability of AI models in forecasting, as well as the use of ensemble learning techniques for improved performance. It focuses on both theoretical concepts and practical applications and offers valuable insights for researchers, practitioners, and professionals mainly in the field of time series forecasting with AI. To prove the applicability of AI, the editors asked the ChatGPT to prepare this Preface, which is reproduced here with insignificant editorial changes, including the description of each one of its 14 chapters.

**Chapter 1:** In this chapter, the authors debate whether AI is approaching human intelligence or is fundamentally different. They explore the achievements of AI in comparison with human intelligence and highlight the complementary nature of these two forms of intelligence. Furthermore, the chapter discusses the current capabilities and future challenges of AI, including the feasibility of Artificial General Intelligence (AGI) and the potential for augmenting human intelligence through Intelligence Augmentation (IA).

**Chapter 2:** This chapter delves into the economic implications of AI and how expectations about the future of AI may shape decision-making. The author explores the possibility that individuals may alter their savings and investment behaviors based on their beliefs about extreme wealth or catastrophic outcomes resulting from future AI developments. The author also discusses the potential for conflict arising from expectations about the military value of AI and the importance of economies of scale in AI development. Additionally, the chapter examines the potential impact of AI-related economic booms on borrowing and public policy decisions.

**Chapter 3:** This chapter focuses on time series forecasting, a critical application of AI in various domains. The author provides an overview of the key advances in time series forecasting methods, ranging from traditional statistical techniques to sophisticated machine learning and deep learning methods. He discusses the advantages and drawbacks of different methods and highlights the conditions under which these methods are expected to perform better. The author also proposes directions for future research to further improve the accuracy and applicability of time series forecasting methods.

**Chapter 4:** In this chapter, the authors cover the challenges of forecasting demand for new products, a crucial task for businesses given the high stakes involved in product launches. They discuss the complex and dynamic nature of demand forecasting in the context of economic competition, changing customer expectations, and emerging technologies. They highlight the high failure rate of new launches and the importance of accurate demand forecasts for decision-making. The chapter emphasizes the need for robust and accurate demand forecasting methods to mitigate risks and optimize business outcomes. It also reviews several case studies that show how machine learning can improve the accuracy of new product forecasts.

**Chapter 5:** This chapter provides insights into the emerging field of global forecasting models, which have shown promising results in forecasting competitions and real-world applications. The author discusses the value of global models in the context of Big Data and how they outperform traditional univariate models when dealing with large collections of related time series. He also highlights the data preparation steps for fitting global models and provides a brief history of their evolution. The chapter concludes with an overview of open-source frameworks available for implementing global models.

**Chapter 6:** This chapter explores how large quantities of data can be leveraged to improve the forecasting accuracy of AI models. The author discusses the challenges and advantages of using AI models for time series forecasting, highlighting the universal applicability of global models and the statistical aspects of cross-learning. The chapter concludes with recommendations for practitioners to enhance the performance of AI forecasting models by tuning both the models and the data sets at hand.

**Chapter 7:** In this chapter, the authors examine what is known as concept drift, which refers to the changes in the underlying data distribution over time, and its negative effect on AI models. They highlight the challenges of concept drift in various domains and review existing methods for handling it, such as adaptive weighting, to provide insights into their strengths and limitations. The authors suggest new ways for handling concept drift in global machine learning models and make suggestions for future research in the field.

**Chapter 8:** This chapter focuses on the combination of forecasts produced by ensembles of feed-forward neural networks for time series forecasting. It discusses the benefits of using forecast combinations, such as improved accuracy and robustness, and the challenges associated with using neural networks, such as their stochastic nature and large number of hyperparameters. The chapter empirically evaluates the performance of individual models and ensembles of models using data sets from the M4 competition, and finds that ensembling neural networks with different initializations and hyperparameters can significantly improve forecasting performance, but at the cost of increased computational time.

**Chapter 9:** This chapter explores the growing interest in time series forecasting with meta-learning, which is a promising method for automatic model selection and combination when dealing with large numbers

of time series. The chapter reviews the current development of meta-learning methods in time series forecasting, summarizes a general meta-learning framework, and discusses the key elements of establishing an effective meta-learning system. It also introduces a python library that aims to make meta-learning available to researchers and practitioners in a unified, easy-to-use framework. The chapter concludes with experimental evaluations of the library on two open-source data sets, showing promising performance of meta-learning in time series forecasting across various disciplines, and offers suggestions for further academic research in this area.

**Chapter 10:** This chapter describes state-of-the-art feature-based methods for forecasting in complex domains, such as economics, where the forecasting performance of different methods varies depending on the nature of the time series. It covers feature-based model selection and combination approaches, with references to open-source software implementations.

**Chapter 11:** This chapter focuses on demand forecasting in the online fashion industry, which presents unique challenges related to large data volumes, irregularity, high turnover in the catalog, and the fixed inventory assumption. The authors elaborate on the data and modeling approach they used to forecast demand using deep learning, highlighting the effectiveness of the proposed method.

**Chapter 12:** This chapter advocates for the integration of forecasting and optimization methods in operations research, as they are widely used in academia and practice in order to deal with uncertainties and make informed decisions. It explores problems that require both forecasting and optimization and discusses the nature of their relationship and potential for integration.

**Chapter 13:** This chapter presents a novel method for monetary policy analysis and inflation forecasting, called LSTVAR-ANN, which imposes different economic dynamics during different periods of the business cycle. It provides insights into the impact of monetary policy on different economic periods, components of the business cycle, and inflation forecasts.

**Chapter 14:** This chapter introduces the forecast value added (FVA) framework as an alternative method for assessing forecasting performance and properly evaluating the advances of AI forecasting models.

This book will inspire researchers, practitioners, and professionals to broaden and deepen their knowledge of the field, discover unique techniques, and share ideas.

Finally, to show the diversity of AI's applications, we would like to end this Preface with the following poem written by ChatGPT about this book.

*AI Shaping Tomorrow's World  
A book that unveils AI's might,  
Forecasts a future, bold and bright.  
From history's lessons, we reflect,  
Ethical AI, we must perfect.  
Machine learning, a power to harness,  
Forecasting trends, a valuable harness.  
Solving challenges, with data and insight,  
AI's potential, a guiding light.*

Eragny, France  
Engomi, Cyprus  
Zografou, Greece

Mohsen Hamoudia  
Spyros Makridakis  
Evangelos Spiliotis

## ACKNOWLEDGMENTS

We extend our sincere gratitude to all the authors of this book for their outstanding contributions, dedication, patience, and flexibility throughout the editing process. We are also grateful to the reviewers for their invaluable feedback and comments that helped us ensure the high quality of each chapter. Additionally, we would like to express our appreciation to our forecasting colleagues and friends from various universities and organizations around the world for their insightful comments and suggestions.

We are deeply grateful to our families for their unwavering support and understanding throughout the project's duration. Their cooperation and encouragement were instrumental in bringing this book to fruition.

We warmly thank Keith Ord, Professor Emeritus at McDonough School of Business, Georgetown University who kindly wrote the Foreword of this book.

Finally, we would like to acknowledge the support of Palgrave and Springer for accepting and publishing the manuscript into Palgrave Advances in the Economics of Innovation and Technology. We extend our sincere thanks to Ellie Duncan, Ashwini Elango, Susan Westendorf, Matthew Savin, Albert N. Link, Bronwyn Geyer, and Meera Seth for their assistance in the realization of this book.

# CONTENTS

## Part I Artificial Intelligence: Present and Future

- |   |   |    |
|---|---|----|
| 1 | <b>Human Intelligence (HI) Versus Artificial Intelligence (AI) and Intelligence Augmentation (IA)</b> | 3  |
|   | Spyros Makridakis and Antonis Polemitis   |    |
| 2 | <b>Expecting the Future: How AI's Potential Performance Will Shape Current Behavior</b>               | 31 |
|   | James D. Miller   |    |

## Part II The Status of Machine Learning Methods for Time Series and New Product Forecasting

- |   |   |    |
|---|---|----|
| 3 | <b>Time Series Forecasting with Statistical, Machine Learning, and Deep Learning Methods: Past, Present, and Future</b> | 49 |
|   | Evangelos Spiliotis   |    |
| 4 | <b>Machine Learning for New Product Forecasting</b>   | 77 |
|   | Mohsen Hamoudia and Lawrence Vanston  |    |

## Part III Global Forecasting Models

- |   |  |     |
|---|--|-----|
| 5 | <b>Forecasting with Big Data Using Global Forecasting Models</b> | 107 |
|   | Kasun Bandara  |     |

<b>6 How to Leverage Data for Time Series Forecasting with Artificial Intelligence Models: Illustrations and Guidelines for Cross-Learning</b>	<b>123</b>
Pablo Montero-Manso	
<b>7 Handling Concept Drift in Global Time Series Forecasting</b>	<b>163</b>
Ziyi Liu, Rakshitha Godahewa, Kasun Bandara, and Christoph Bergmeir	
<b>8 Neural Network Ensembles for Univariate Time Series Forecasting</b>	<b>191</b>
Artemios-Anargyros Semenoglou, Evangelos Spiliotis, and Vassilios Assimakopoulos	
<b>Part IV Meta-Learning and Feature-Based Forecasting</b>	
<b>9 Large-Scale Time Series Forecasting with Meta-Learning</b>	<b>221</b>
Shaohui Ma and Robert Fildes	
<b>10 Forecasting Large Collections of Time Series: Feature-Based Methods</b>	<b>251</b>
Li Li, Feng Li, and Yanfei Kang	
<b>Part V Special Applications</b>	
<b>11 Deep Learning Based Forecasting: A Case Study from the Online Fashion Industry</b>	<b>279</b>
Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma, and Tim Januschowski	
<b>12 The Intersection of Machine Learning with Forecasting and Optimisation: Theory and Applications</b>	<b>313</b>
Mahdi Abolghasemi	
<b>13 Enhanced Forecasting with LSTVAR-ANN Hybrid Model: Application in Monetary Policy and Inflation Forecasting</b>	<b>341</b>
Michał Chojnowski	

<b>14 The FVA Framework for Evaluating Forecasting Performance</b>	<b>373</b>
Michael Gilliland	
<b>Author Index</b>	<b>385</b>
<b>Subject Index</b>	<b>397</b>

# EDITORS AND CONTRIBUTORS

## About the Editors

**Mohsen Hamoudia** is CEO since 2020 of PREDICONSULT (Data and Predictive Analytics), Paris. He has held positions in Orange and France Telecom, *inter alia*, as Head of New Products and Planning Division (1993–1997), Head of Planning and Forecasting Division in France Telecom Long Distance (1997–2001), Head of Strategic Marketing of Large Projects Division (2001–2013), and VP Head of Strategy and Market Intelligence of Large Accounts Division within Orange Business Services, Paris (2013–2019). Mohsen teaches Forecasting and Quantitative Techniques since 1989 at Toulouse Business School, ESDES Business School-Lyon, ISM (Institut Supérieur du Marketing) Paris, and College Polytechnique, Paris. He was a Board member of the International Institute of Forecasters (IIF) from 2009 to 2020 and its past President (2012–2016). He is also a Board member of OrangeFab, the startups acceleration program of Orange. Mohsen’s research interests are broad. His research is primarily focused on empirical aspects of forecasting in air transportation, telecommunications, ICT (Information and Communication Technologies), electronic payment, social networking, and innovation and new technologies. He has authored several book chapters and co-edited with James Alleman and Paul N. Rappoport “Applied Economics in the Digital Era”, Palgrave Macmillan, 2020.

He has written several articles in the *International Journal of Forecasting*, *Technology Forecasting and Social Change*, and *Eurasian Business Review* and was an Editorial Board Member of *Telecommunications Policy*, Elsevier.

**Spyros Makridakis** is a Professor at the University of Nicosia, Cyprus, where is a Director of its Institute For the Future (IFF), and the founder of the Makridakis Open Forecasting Center (MOFC). He is also an Emeritus Professor at INSEAD, France he joined in 1970. He has authored, or co-authored, twenty-seven books and special issues and more than 360 articles. His book “Forecasting Methods for Management”, 5th ed. (Wiley) has been translated in twelve languages and sold more than 120,000 copies while his book “Forecasting: Methods and Applications”, 3rd ed. (Wiley) has received more than 6,800 citations. Spyros was the founding Editor-in-Chief of the *Journal of Forecasting* and the *International Journal of Forecasting* and the co-founder of the International Institute of Forecasters (IIF). He is also the organizer of the renowned M (Makridakis) competitions that for the last 40 years have fundamentally influenced the theory and practice of forecasting.

**Evangelos Spiliotis** is a Research Fellow at the Forecasting & Strategy Unit, National Technical University of Athens (NTUA), Greece, where he also serves as Coordinator. He holds a Diploma (MEng) in Electrical and Computer Engineering and a DEng in Forecasting Support Systems, both granted by NTUA. His research interests include time series forecasting, decision support systems, machine learning, optimization, and energy efficiency and conservation. He has conducted research and development on tools for management support in many projects, funded by the European Commission and enterprises of the banking, energy, retail, and transportation industries. Evangelos is currently an Associate Editor for the *International Journal of Forecasting* and a Machine Learning and AI Column Editor of the *Foresight: The International Journal of Applied Forecasting*. He has co-organized the M4, M5, and M6 forecasting competitions and published his work in prestigious conferences and academic journals, such as *International Journal of Forecasting*, *European Journal of Operational Research*, *Machine Learning*, *Neural Networks*, *Pattern Recognition*, and *Applied Soft Computing*.

## Contributors

**Mahdi Abolghasemi** is a lecturer in Data Science at The University of Queensland, Australia. He has a multi-disciplinary background in engineering, business, statistics, and machine learning. His research interests/expertise are in time series forecasting, predictive analytics, data science, and machine learning with applications in supply chain management and renewable energies optimization. He has published over 40 articles and technical reports and presented at numerous national and international conferences. Mahdi's research is closely tied to evidence-based analytics for real-world problems. Through his research he has consulted dozens of companies in Australia, Europe, and the Middle East and developed several analytical models to improve their supply chain networks and analytical capabilities.

**Vassilios Assimakopoulos** is a professor of Forecasting Systems at the School of Electrical and Computer Engineering of the National Technical University of Athens (NTUA), Greece. He has worked extensively and conducted research on innovative tools for management support in numerous projects, funded by the European Commission, the Greek Government, and enterprises of various industries. Vassilios specializes in forecasting, strategic management, design and development of information systems, as well as business resource management. He is the author of more than 100 original publications, the creator of many popular time series forecasting methods, such as Theta and ADIDA, and has co-organized the M4, M5, and M6 forecasting competitions.

**Kasun Bandara** is a Forecasting and Analytics Analyst at EnergyAustralia and an Honorary Research Fellow in the School of Computing and Information Systems at University of Melbourne. Kasun holds a PhD in Computer Science from the Monash University, Australia, and a B.Sc. honors degree in Computer Science from University of Colombo School of Computing, Sri Lanka. Kasun has developed machine learning based forecasting architectures in collaboration with research scientists from Uber, Walmart, and Facebook and has published his research work in journals, such as *IEEE Transactions on Neural Networks and Learning Systems*, *Pattern Recognition*, and *International Journal of Forecasting*, as well as in conferences, such as *PAKDD*, *IJCNN*, and *ICONIP*. Kasun is a reviewer for leading machine learning conferences, such as *ICML*,

*NeurIPS*, and *ICLR*. In addition to research work, Kasun regularly participates in forecasting related competitions. Few of his notable recognitions are IEEE CIS Energy Forecasting Competition (4th Position), Fuzz-IEEE Competition on Explainable Energy Prediction (2nd Position), M5 Forecasting Competition (Kaggle Gold Medalist), Air-Liquide Future Ready Data Challenge (4th position), and M6 Forecasting Competition (11th Position).

**Christoph Bergmeir** is a Senior Lecturer in Data Science and Artificial Intelligence in the Department of Data Science and Artificial Intelligence at Monash University, Australia. He works as a Data Scientist in a variety of projects with external partners in diverse sectors, such as sustainable energy and supply chain. He has led teams that have delivered systems for short-term power production forecasting of wind and solar farms, and energy price forecasting. Christoph holds a PhD in Computer Science from the University of Granada, Spain and an M.Sc. degree in Computer Science from the University of Ulm, Germany. He has over 5,000 citations and an h-index of 28. He has received more than \$2.7 million in external research funding. Four of his publications on time series forecasting over the last years have been Clarivate Web of Science Highly Cited Papers (top 1% of their research field).

**Stefan Birr** is a Senior Applied Scientist working on time series forecasting at Zalando SE. Before joining Zalando he obtained his doctoral degree researching “Dynamic Dependencies in Time Series”, and had the Lead Expert Role in time series analysis at E.ON Digital Technology.

**Michał Chojnowski** is an Economic Analysis Manager in the chemical industry. He has vast experience in demand forecasting and economic analysis, working on markets such as apparel, winery, automotive, and pharmacy. His expertise lies in non-linear forecasting including Business Cycle dynamics. Michael holds a BA in Economics from Warsaw School of Economics, Poland and an MSc in Econometrics and Mathematical Economics from Tilburg University, Netherlands. Currently, he is a PhD candidate in Warsaw School of Economics. His research interests includes non-linear macroeconomic forecasting, artificial neural networks application in inflation forecasting, and market sentiment analysis. His works were presented at International Symposium on Forecasting and CIRET Institute and published in journals such as *Econometric Research in Finance*.

**Robert Fildes** is a Distinguished Professor in the Management School, Lancaster University, Founding Director of the Centre for Marketing Analytics and Forecasting, and he recently co-authored “Principles of Business Forecasting”, Wessex. In 1980 with Spyros Makridakis and J. Scott Armstrong was a founding director of the International Institute of Forecasters (IIF): the Institute has led the way in establishing a rigorous approach to the evaluation of new forecasting approaches through its journal, academic conferences, and workshops. In 1982 he contributed to one of the first major forecasting competitions, led by Makridakis, the M competition, and he has continued to research the comparative evaluation of different forecasting methods, particularly applied to retailing. Most recently he has worked with Shaohui Ma on major retail bases using machine learning methods. In addition to novel methodological ideas, such as those explained in his present chapter on meta-learning, he has focused on overcoming the barriers to effective implementation of improved forecasting procedures and systems: in a nutshell, getting improvements into practice.

**Michael Gilliland** is Editor-in-Chief of *Foresight: The International Journal of Applied Forecasting*, taking this position in 2022 after ten years on Foresight’s editorial staff. He retired from SAS in 2021, where he held positions in marketing and product management for SAS forecasting software, and prior to this spent 19 years in forecasting, supply chain, and consulting positions in the food, consumer electronics, and apparel industries. Michael is author of “The Business Forecasting Deal” (2010), and principal editor of “Business Forecasting: Practical Problems and Solutions” (2015) and “Business Forecasting: The Emerging Role of Artificial Intelligence and Machine Learning” (2021). He also edited “Forecasting with SAS®: Special Collection” (2020) and for twelve years wrote *The Business Forecasting Deal* blog. In 2017 Michael received the Lifetime Achievement Award from the Institute of Business Forecasting, and in 2021 his paper “Forecast Value Added: A Reality Check on Forecasting Practices” was inducted into the Foresight Hall of Fame. Michael holds a BA in Philosophy from Michigan State University, and master’s degrees in Philosophy and Mathematical Sciences from Johns Hopkins University. He is interested in issues relating to forecasting process and performance, such as worst practices and Forecast Value Added (FVA) analysis, and in applying research findings to real-life improvement in business forecasting.

**Rakshitha Godahewa** is a Postdoctoral Research Fellow at Monash University, Australia. She also completed her PhD at Monash University. Her research focuses on time series forecasting using global modeling and ensembling where the findings are published in top-quality outlets including *Neural Information Processing Systems* and *Machine Learning*. In 2022, Rakshitha participated in the 9th Heidelberg Laureate Forum and was recognized as one of the top 200 young researchers in the world. She is also among the top performers in prestigious forecasting competitions including the M5 and M6 competitions. Rakshitha holds a Degree of Bachelor of the Science (Honors) in Engineering specialized in Computer Science and Engineering from University of Moratuwa, Sri Lanka, where she scored a First Class and was recognized as one of the top four students in the department. She also holds the British Computer Society (BCS) degree from BCS, UK. Prior to commencing her post-graduate studies at Monash University, she worked in the IT industry as a Senior Software Engineer for nearly three years.

**Tim Januschowski** is the director of pricing platform at Zalando SE with prior roles at Amazon and SAP. He regularly publishes scientifically on forecasting and teaches professionals and students at universities, most recently TU Munich, Germany. His community service includes being a director at the International Institute of Forecasters (IIF).

**Yanfei Kang** is an Associate Professor of Statistics in the School of Economics and Management at Beihang University, China. Prior to that, she was a Senior R&D Engineer in the Big Data Group of Baidu Inc. Yanfei obtained her Ph.D. in Applied and Computational Mathematics at Monash University in 2014, and served at Monash as a postdoctoral research fellow through 2015. Her research interests include time series forecasting, time series visualization, statistical computing, and machine learning.

**Manuel Kunz** is an Applied Science Manager and former Applied Scientist in the Zalando Pricing Department. He has a background in computer science and machine learning. Before joining Zalando in 2015, he worked as a Software Engineer for a technology consultancy company.

**Feng Li** is an Associate Professor of Statistics in the School of Statistics and Mathematics at the Central University of Finance and Economics in Beijing, China. He earned his Ph.D. in Statistics from Stockholm University, Sweden in 2013. Feng's research interests include Bayesian statistics,

econometrics and forecasting, as well as distributed learning. He develops highly scalable algorithms and software for solving real business problems.

**Li Li** is a Ph.D. student in statistics from the School of Economics and Management, Beihang University, China. She has published many academic papers in journals, such as *International Journal of Forecasting* and *International Journal of Production Research*. Her research interests include time series forecasting and statistical calculation.

**Ziyi Liu** is a first-year PhD student at Monash University, Australia. He holds a Certificate in Quantitative Finance (CQF). His PhD research focuses on black-box optimization via the Bayesian framework for statistical learning and utilizing information-theoretic techniques such as minimum message length and minimum description length. Models of interest include generalized additive models, sets of rules, and deep neural networks. Ziyi holds a Master's Degree in Data Science from Monash University. His master program research area is time series forecasting, especially for time series with concept drift. He received high distinctions in his master's program.

**Lei Ma** is an Applied Scientist at Zalando SE. He received a Ph.D. in Physics in 2018 from the University of New Mexico. He has been working on neutrino physics and deep learning applications in pricing.

**Shaohui Ma** is a Professor of Management Science at Nanjing Audit University, China. He also serves as the Director of the Big Data Management Department and the Management Science Research Center. He earned his M.Sc. and Ph.D. degrees in Management Science from Tianjin University, China. His research focuses on big data-driven business analytics and decision optimization, with particular emphasis on business forecasting, customer relationship management, and decision support systems. He has successfully completed four research projects supported by the National Natural Science Foundation of China and has published over 40 academic papers in top-tier journals such as *European Journal of Operational Research*, *International Journal of Forecasting*, *Information Sciences*, and *Marketing Letters*. In addition, he has authored several books and received awards and honors for his contributions to the field of management science.

**Pablo Montero-Manso** is a Lecturer at the Discipline of Business Analytics, University of Sydney, Australia. He holds a PhD in Statistics

from the University of A Coruña, Galicia, Spain. He researches machine learning, artificial intelligence, and statistical tools for time series analysis, focusing on forecasting, classification, clustering, and visualization. Pablo has developed predictive models that resulted in award-winning performance in major forecasting competitions (M4 and M6) and have been adopted by the industry. During the COVID-19 pandemic, he contributed with highly accurate models and predictions of the evolution of the pandemic that were part of the decision-making process in Australia, Spain, and the European Union. Pablo is a member of the advisory board of the WHY project, analyzing energy consumption of European households for policy making. He is an author and contributor to several open-source tools and open datasets in Python and R, including the popular TSclust package for time series clustering.

**James D. Miller** is a Professor of Economics at Smith College, Massachusetts. He has a J.D. from Stanford, California and a Ph.D. in economics from the University of Chicago, Illinois. His academic interests include game theory and the technological singularity.

**Antonis Polemitis** currently serves as the Chief Executive Officer of the University of Nicosia and EDEX, as a Board member of EDEX and UNICAF, and as a member of the Council of the University of Nicosia. The University of Nicosia (UNIC) serves over 14,000 students, along with over 18,000 additional students in its affiliated academic institutions. UNIC is the largest university in Cyprus and is the largest English language university in southern Europe. Antonis helped found the world-leading Digital Currency / Blockchain Initiative at the University of Nicosia, co-taught the first university cryptocurrency course in the world, and is regularly quoted as an expert on cryptocurrency issues. He was a member of the national committee that designed the blockchain strategy for Cyprus. Antonis is the managing partner of Ledra Capital where he led early-stage investments in, or software development of, Software-as-a-Service platforms in the areas of higher education, cryptocurrency, online video publishing, and legal research. He was previously a principal at ACG Capital, a privately held multi-billion dollar investment firm and a partner based in New York and London in the private equity practice of Mercer Management Consulting (now Oliver Wyman), one of the world's leading strategy consultancies.

**Mones Raslan** is a deep learning researcher with a background in mathematics. After his studies and PhD at TU Berlin with focus on the theory of neural networks, he joined Zalando's Pricing Platform in 2022 as an Applied Scientist.

**Artemios-Anargyros Semenoglou** is a PhD student and research associate at the Forecasting & Strategy Unit of the National Technical University of Athens (NTUA), Greece. He completed his Bachelor's degree in Electrical and Computer Engineering at the NTUA in 2017. His primary research interests include time series forecasting, data analytics, and the application of machine learning in operational research. As a data scientist, he has worked on numerous projects across various business domains developing predictive machine learning solutions. His academic and research work is mainly focused on the development and application of global neural networks in general-purpose, univariate time series forecasting.

**Lawrence Vanston** is an internationally-recognized authority on technological forecasting. His focus is on the future adoption of new products, services, and technologies, the nature of technological advance, and the impacts of technology on business and society. As president of Technology Futures, Inc. (TFI), he has conducted and managed future-looking studies for the communications and other high-tech industries for over forty years. His current research interests include broadband, wireless, video, AI, extended reality, and emerging technologies. He is also founder and executive director of the non-profit, TFI Projects. As an expert on the impact of technology change on asset valuation, Lawrence has served as an expert witness in regulatory, tax, and other proceedings. He has been the principal author of numerous reports commissioned by government agencies, industry organizations, and individual companies, including all of the major U.S. broadband and wireless providers. The Communications Technology Forecasting Group (CTFG), currently consisting of AT&T, Charter/Spectrum, Comcast, and Cox, has been actively supporting TFI research since 1985. Lawrence is a frequent speaker at the *International Symposium on Forecasting*, the *Wichita Program for Appraisal for Ad Valorem Taxation*, and the *TFI Technology Conference*, which he launched in 2006. He is a frequent contributor to *Foresight: the International Journal of Applied Forecasting* and serves on its advisory board. Before joining Technology Futures in 1984, Lawrence was with Bell Labs and Bellcore in network planning. He holds a B.A. in

Government (1975) and an M.S. (1977) and Ph.D. (1979) in Operations Research and Industrial Engineering, all from the University of Texas, Austin.

## ABBREVIATIONS

AdaBoost	Adaptive Boosting
ADL	Autoregressive Distributed Lag
ADLP	ADL with data Pooling
ADP	Adaptive Dynamic Programming
AFNNC	Adaptive Fuzzy-Neural-Network Controller
AGI	Artificial General Intelligence
AI	Artificial Intelligence
aML	automatic Machine Learning
ANC	Adaptive Neural Control
ANNs	Artificial Neural Networks
APE	Average Prediction Error
API	Application Programming Interface
AR	Augmented Reality
AR	Auto Regressive
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	AutoRegressive Integrated Moving Average with external variables
ASI	Artificial Super Intelligence
AveRelMAE	Average Relative MAE
AVs	Autonomous Vehicles
BBIs	Brain to Brain Interfaces
BBN	Bayesian Belief Network
BCA	Bounded Component Analysis
BCI	Direct Brain to Computer Interface
BCNs	Boolean Control Networks
BIC	Schwarz Information Criteria

BMAL	Batch Mode Active Learning
BMs	Bayesian Models
BN	Bayesian Network
BNN	Bayesian Neural Network
BRNN	Bidirectional Recurrent Neural Network
CALA	Continuous Action-set Learning Automata
CARTs	Classification And Regression Trees
CB	CatBoost
CD	Coordinate Descent
CDNs	Complex Dynamical Networks
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DC3	Deep Constraint Completion and Correction
DCGAN	Deep Convolutional Generative Adversarial Network
DL	Deep Learning
DMNNs	Delayed Memristive Neural Networks
DNNs	Discrete-time Neural Networks
DRNNs	Delayed Recurrent Neural Networks
DSS	Decision Support System
EANNs	Evolutionary Artificial Neural Networks
ECW	Error Contribution Weighting
EEC	Estimation Error Covariance
EIA	Error Interaction Approach
ELMs	Extreme Learning Machines
ES-RNN	Exponential Smoothing-Recurrent Neural Network
ETS	Error Trend and Seasonality
FFORMA	Feature-based FOREcast Model Averaging
FFORMS	Feature-based FOREcast Model Selection
FVA	Forecast Value Added
GBRTs	Gradient Boosting Regression Trees
GBTs	Gradient Boosted Trees
GDW	Gradient Descent Weighting
GFMs	Global Forecasting Models
GRNN	Generalized Regression Neural Network
HCA	Hierarchical Clustering Analysis
HI	Human Intelligence
HRL	Hierarchical Reinforcement Learning
IA	Intelligence Augmentation
IRF	Impulse Response Function
ITL	Information Theoretic Learning
KNNR	K-Nearest Neighbors Regression
KPI	Key Performance Indicator
KRLS	Kernel Recursive Least-Squares

LapSVM	Laplacian Support Vector Machine
LGBM	Light Gradient Machine
LODL	Locally Optimised Decision Loss
LSSVMs	Least-Squares Support Vector Machines
LSTM	Long Short-Term Memory
LSTVAR	Logistic Smooth Transition Vector AutoRegressive
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MARS	Multivariate Adaptive Regression Splines
MART	Multiple Additive Regression Tree
MASE	The Mean Absolute Scaled Error
MBMI	Mind/Body/Machine Interface
MCNNs	Memristor-based Cellular Neural Networks
MDPs	Markov Decision Processes
MDRNN	Multidimensional Recurrent Neural Network
MEKA	Multiple Extended Kalman Algorithm
Meta-GLAR	Meta Global-Local Auto-Regression
MFAC	Model-Free Adaptive Control
MIMO	Multi-Input Multi-Output
MIP	Mixed Integer Programming
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MNNs	Memristor-based Neural Networks
MRNNs	Memristor-based Recurrent Neural Networks
MSE	Mean Squared Error
MTRNN	Multiple Timescale Recurrent Neural Network
NARMA	Non-linear Autoregressive Moving Average
N-BEATS	Neural Basis Expansion Analysis for Time Series
NCSTAR	Neural Coefficient Smooth Transition Autoregressive
NLP	Natural Language Processing
NLU	Natural Language Understanding
NN	Neural Network
NTM	Neural Turing Machine
PB	Pseudocost Branching
PCA	Principal Component Analysis
PCVM	Probabilistic Classification Vector Machine
PID	Partial Information Decomposition
PLSR	Partial Least Squares Regression
PMI	Pointwise Mutual Information
PNN	Probabilistic Neural Network
QRF	Quantile Regression Forest
RADP	Robust Adaptive Dynamic Programming

RBM	Restricted Boltzman Machines
RCD	Recurring Concept Drift
RCNNs	Recurrent Convolutional Neural Networks
ReLU	Rectified Linear Unit
REPTree	Reduced Error Pruning Tree
RF	Random Forest
RL	Reinforcement Learning
RLFMs	Regression Based Latent Factors
RMSE	Root Mean Squared Error
RMSProp	Root Mean Squared Propagation
RMSSE	Root Mean Squared Scaled Error
RNNLM	Recurrent Neural Network Language Model
RNNs	Recurrent Neural Networks
RSONFIN	Recurrent Self-Organizing Neural Fuzzy Inference Network
RSS	Residual Sum of Squares
RTRL	Real-Time Recurrent Learning
SAA	Sample Average Approximation
SAE	Social Adaptive Ensemble
SB	Strong Branching
scMAE	scaled Mean Absolute Error
SDL	Supervised Descriptor Learning
SES	Simple Exponential Smoothing
SGBoost	Stochastic Gradient Boosting
SGD	Stochastic Gradient Descent
SKUs	Stock Keeping Units
SLDR	Supervised Linear Dimension Reduction
sMAPE	symmetric MAPE
SNPOM	Structured Nonlinear Parameter Optimization Method
SNRFs	Symbolic Network Reliability Functions
SOM	Self-Organizing Map
SPO	Smart Predict and Optimise
SSVM	Smooth Support Vector Machine
SVA	Stochastic Value Added
SVM	Support Vector Machine
SVOR	Support Vector Ordinal Regression
SVR	Support Vector Regression
SVRG	Stochastic Variance Reduced Gradient
TA-SVM	Time-adaptive Support Vector Machine
TPE	Total Percentage Error
TRPO	Trust Region Policy Optimization
TWSVC	Twin Support Vector Clustering
VARIMA	Vector ARIMA
wMAPE	weighted MAPE
XGB	XGBoost

## LIST OF FIGURES

Fig. 4.1	1995 High-definition television forecast (Vanston et al., 1995) ( <i>Source</i> Technology Futures, Inc.)	82
Fig. 4.2	Forecasting techniques for new products (Hamoudia, 2021)	83
Fig. 5.1	Sales demand of four different products over a six months period, extracted from the M5 forecasting competition dataset, which is based on the daily product demand data from <i>Walmart</i>	109
Fig. 5.2	An example of applying the moving window approach to a time series from the M5 forecasting competition dataset	110
Fig. 5.3	The input-output window matrix after applying moving window approach to a time series length of $k$ . Here, the size of the input window is $n$ and the output window size is 1	112

Fig. 6.1	a Predictions of a sigmoid model fit to data for the COVID-19 infections in Italy using data up to 28 March 2020. The vertical striped line indicates the date on which the forecast is made, and the model is fit using only data before that date. Left panel represents daily cumulative cases, and right panel represents new infections per day. Long-term predictions are very off the mark, arguably because the models are very unreliable before the peak of the series. b Predictions of the same model, based on data on the following day, 29 March 2020. Predictions are much more accurate, and we can identify the pre-peak/post-peak behavior of the model. Fitting the same model on subsequent dates also produces reliable predictions.	157
Fig. 6.2	a Predictions of a sigmoid model fit to data for the COVID-19 infections in the UK using data up to 29 March 2020, when Italy had already peaked. Since the UK has not peaked, the predictions of the sigmoid model are not accurate. b Predictions of a local (not cross-learned) linear autoregressive model for the UK using the same data. The performance of a local linear autoregressive model is even more inaccurate than those of the more fundamental sigmoid model, since the c predictions for a cross-learned linear autoregressive model that includes the time series for Italy in the set. The cross-learned model has effectively transferred the pre-peak/post-peak behavior from Italy to the UK.	159
Fig. 7.1	A visualisation of sudden, gradual and incremental concept drift types	167
Fig. 7.2	Performance of all models in terms of MAE (left) and RMSE (right), across the series showing sudden concept drift with different drift points	185
Fig. 7.3	Performance of all models in terms of MAE (left) and RMSE (right), across the series showing incremental concept drift with different drift lengths	186

Fig. 8.1	Distribution of MASE forecasting errors of 200 randomly initialized base NNs models and 200 ensembles ensembles consisting of 10 randomly initialized base NNs each, for the cases of <i>M4_Yearly</i> , <i>M4_Quarterly</i> and <i>M4_Monthly</i> sets of series. The dashed horizontal line represents the forecasting accuracy of ETS. The performance of ETS in <i>M4_Yearly</i> is not shown as it falls outside the axis range (MASE: 3.444)	205
Fig. 8.2	Forecasting performance (MASE) of ensembles of NNs, for different ensembles sizes, ranging from 1 up to 200 models per ensemble, for the cases of <i>M4_Yearly</i> , <i>M4_Quarterly</i> , and <i>M4_Monthly</i> sets of series	207
Fig. 8.3	Distribution of MASE forecasting errors of 10 base NNs trained with different loss functions (scMAE, MAPE, sMAPE, and MASE), for the cases of <i>M4_Yearly</i> , <i>M4_Quarterly</i> , and <i>M4_Monthly</i> sets of series. “X” marks the forecasting accuracy of the respective loss function-specific ensemble. The red-dashed horizontal line represents the forecasting accuracy of the ensemble that combines the forecasts of all available models, using all four training losses. The black-dashed horizontal line represents the forecasting accuracy of ETS. The performance of ETS in <i>M4_Yearly</i> is not shown as it falls outside the axis range (MASE: 3.444)	209
Fig. 8.4	Distribution of sMAPE forecasting errors of 10 base NNs with different input layer sizes (from 1 up to 7 times the corresponding forecasting horizon), for the cases of <i>M4_Yearly</i> , <i>M4_Quarterly</i> , and <i>M4_Monthly</i> sets of series. “X” marks the forecasting accuracy of the ensemble combining networks with a particular input layer size. The red-dashed horizontal line represents the forecasting accuracy of the ensemble that combines the forecasts of all available models of different input layer sizes. The black-dashed horizontal line represents the forecasting accuracy of ETS. The performance of ETS in <i>M4_Yearly</i> is not shown as it falls outside the axis range (MASE: 3.444)	210
Fig. 9.1	A general meta-learning framework for time series forecasting	227
Fig. 9.2	Network structure of the ‘MetaComb’	239

Fig. 9.3	Network structure of the ‘MetaCombFeat’	240
Fig. 9.4	Network structure of the ‘MetaSelection’	240
Fig. 9.5	Network structure of ‘MetaLoss’	241
Fig. 10.1	The framework of feature-based methods for forecasting	252
Fig. 10.2	Typical examples of recurrence plots (top row) for time series data with different patterns (bottom row): uncorrelated stochastic data, i.e., white noise (left), a time series with periodicity and chaos (middle), and a time series with periodicity and trend (right)	261
Fig. 11.1	Histogram of demand (in log scale) per article: The histogram shows the heavy-tailed demand distribution. On the x-axis is demand starting from 0, on the y-axis the frequency of the demand occurrence over one time unit	282
Fig. 11.2	Examples for complications occurring in practice: The dashed orange line gives discount level (the higher, the more discount), blue is observed (ground truth) demand and green with markers is the forecast. Note that everything left to the first green dot is the available demand history. We have hence articles with little to no history, see Fig. 11.2a and b as well as sparse time series, Fig. 11.2c	283
Fig. 11.3	Example for a challenging stock situation: The red dashed line represents stock levels over time. The article was out of stock for a longer time period until it was replenished in April 2021. It can be seen that the ground truth is highly correlated with stock availability of an article which provides additional modelling challenges as discussed in section “Sales to Demand Translation”	284
Fig. 11.4	Sales to demand translation for an article with 5 sizes: In <b>a</b> , historical sales observations of an article over weeks with all sizes available are aggregated. Based on <b>a</b> , we obtain in <b>b</b> , the article’s empirical probability distribution over sizes. In <b>c</b> , the weekly demand of an article with missing sizes L and XL is illustrated. The unobserved demand for L and XL is inferred in <b>d</b> , given the observed demand of the other sizes for that week and the empirical distribution computed in <b>b</b>	285

Fig. 11.5	Demand vs sales for a single article. In weeks with full availability, sales and demand overlaps. If a size or the full article becomes unavailable, demand becomes an estimate larger than the observed sales	286
Fig. 11.6	Demand Forecaster Architecture: Encoder handles the observational time series, decoder incorporates the future covariates. Future discounts are directly provided to the output (Monotonic Demand) layer, skipping decoder and encoder	289
Fig. 11.7	Demand and discount time series of an article, illustrating the positive correlation between discount (orange dashed) and demand (blue)	293
Fig. 11.8	Piecewise linear monotonic demand response function as defined in (11.10), x-Axis: the discount level, y-axis: The increase in demand under the applied discount. The $\delta$ values represent the change in demand for a 10% point change in discount	294
Fig. 11.9	Demand model accuracy over a 26 weeks forecast horizon	296
Fig. 11.10	Forecasting pipeline set-up in production	298
Fig. 11.11	Log-log plot of the training-set-size-to-demand-error relationship for three different start dates with constant test set. The dashed lines represent the forecast performance of the naive forecaster (last observation in training horizon)	303
Fig. 11.12	Effect of retraining on accuracy: The plot shows how much accuracy the model loses if not retrained regularly, provided a certain level of randomness introduced by the training process	305
Fig. 13.1	Hidden factors extracted from price-related Google Trends queries ( <i>Note</i> Lines represent values of hidden layer aggregating the employment-related Google Trends queries. Solid-black lines represent sentiment towards oil market; red-dashed line represents sentiment towards gas market; blue-dotted line represents sentiment towards car market. <i>Source</i> Own) (Color figure online)	353

Fig. 13.2	Hidden factors extracted from savings-related Google Trends queries ( <i>Note</i> Lines represent values of hidden layer aggregating the savings-related Google Trends queries. Solid-black lines represent values of the factor related to financial instruments; red-dashed line represents the sentiment towards credit card. <i>Source Own</i> ) (Color figure online)	354
Fig. 13.3	Hidden factors extracted from capital-related Google Trends queries ( <i>Note</i> Lines represent values of hidden layer aggregating the capital-related Google Trends queries. Solid-black lines represent values of investment sentiment; red-dashed line represents the bearish market sentiment; blue-dotted line represents real estate market condition; green-dotdashed line represents IPO opportunities. <i>Source Own</i> ) (Color figure online)	355
Fig. 13.4	Hidden factors extracted from employment-related Google Trends queries ( <i>Note</i> Lines represent values of hidden layer aggregating the employment-related Google Trends queries. Solid-black lines represent values of current employment sentiment; red-dashed line represents career seek effort; blue-dotted line represents job market slowdown; green-dotdashed line represents interest in insurance companies. <i>Source Own</i> ) (Color figure online)	356
Fig. 13.5	Ranks of factors for each expanding window draw ( <i>Note</i> Matrix represents the importance rank of each unobserved factor. The darker the colour, the more important the factor was in a given period. On the y-axis, reader can read the factor ID—for full description the author refers to Table 13.1. <i>Source Own</i> )	358
Fig. 13.6	Transition values: period 2004–2014 ( <i>Note</i> Lines represent the value of the transition function in each period for each LSTVAR-ANN model: Google Trends with supervised time series [solid black], Google Trends without supervised time series [dashed red], PCA with supervised time series [dotted blue], PCA without supervised time series [dotdashed green]. <i>Source Own</i> ) (Color figure online)	359

Fig. 13.7	Transition values: period 2004–2016 ( <i>Note</i> Lines represent the value of the transition function in each period for each LSTVAR-ANN model: Google Trends with supervised time series [solid black], Google Trends without supervised time series [dashed red], PCA with supervised time series [dotted blue], PCA without supervised time series [dotdashed green]. <i>Source Own</i> ) (Color figure online)	360
Fig. 13.8	3D IRF of interest impulse on CPI ( <i>Note</i> Graph represents a 3D-IRF of interest rate on CPI. X-axis represents number of month after the increase in interest rate occurs. Y-axis represents the value of the transition function, identifying the regime in which economy operates. Green colour indicates an increase of prices m/m, red colour a decrease. The more intense the colour, the higher the magnitude of change is recorded given month. <i>Source Own</i> ) (Color figure online)	361
Fig. 13.9	Architecture of sentiment-supervised LSTVAR-ANN hybrid model ( <i>Note</i> Schema represents the architecture of the LSTVAR-ANN model used in this chapter. The top part represents the aggregation part of the Google Trends into unobserved factors. Middle part represents transforming unobserved factors into market sentiment. For “supervised” models, the layer “sentiment-out” is the output layer. The bottom part represents the LSTVAR model. <i>Source Own</i> )	367
Fig. 14.1	Example of a multi-step forecasting process	378

## LIST OF TABLES

Table 1.1	Major available AI applications superior to or approaching HI	9
Table 1.2	Various types of AI versus real life applications	11
Table 4.1	Brief description of the companies and new products (Steenbergen & Mes, 2020)	93
Table 4.2	Predictive performance of DemandForest compared to benchmarks (Steenbergen & Mes, 2020)	94
Table 7.1	Results across sudden, incremental and gradual concept drift types. The best performing models in each group are italicised and the overall best performing models are highlighted in boldface	180
Table 7.2	Results of statistical testing across sudden, incremental and gradual concept drift types	183
Table 8.1	The architecture and the training hyper-parameters of the baseline NNs used in the ensembles, along with the values selected for the purposes of this study, for the <i>M4_Yearly</i> , <i>M4_Quarterly</i> , and <i>M4_Monthly</i> sets of series	204
Table 8.2	Overall forecasting accuracy (MASE) of ensembles of models with different input sizes (from 1 up to 7 times the corresponding forecasting horizon) and loss functions (sMAPE, MASE, MAPE, scMAE), in the <i>M4_Yearly</i> set of series	211

Table 8.3	Overall forecasting accuracy (MASE) of ensembles of models with different input sizes (from 1 up to 7 times the corresponding forecasting horizon) and loss functions (sMAPE, MASE, MAPE, scMAE), in the <i>M4_Quarterly</i> set of series	212
Table 8.4	Overall forecasting accuracy (MASE) of ensembles of models with different input sizes (from 1 up to 7 times the corresponding forecasting horizon) and loss functions (sMAPE, MASE, MAPE, scMAE), in the <i>M4_Monthly</i> set of series	213
Table 9.1	Performance of base forecasters and meta-learners on the M5 dataset	245
Table 9.2	Performance of base forecasters and meta-learners on the KDD2022 dataset	246
Table 10.1	Overview of literature on feature-based forecasting	254
Table 11.1	Examples for covariates available for the forecasting problem at Zalando	286
Table 11.2	Model Performance	300
Table 13.1	Top drivers for unobserved factors	352
Table 13.2	Top 10 important Google Trends queries	357
Table 13.3	RMSE of selected models: pre-COVID-19 scenario	363
Table 13.4	RMSE of selected models COVID-19 scenario	364
Table 13.5	List of used Google Trends queries	368
Table 14.1	Stairstep report with FVA results	376
Table 14.2	Combination of non-value adding steps	381

PART I

---

## Artificial Intelligence: Present and Future



## CHAPTER 1

---

# Human Intelligence (HI) Versus Artificial Intelligence (AI) and Intelligence Augmentation (IA)

*Spyros Makridakis and Antonis Polemitis*

## INTRODUCTION

Artificial Intelligence (AI), as its name implies, is another form of intelligence analogous to our own Human Intelligence (HI). There is agreement that AI has surpassed HI in some specific areas such as games and image recognition, but disagreement on how such superiority is achieved, how fast it would spread to other areas beyond games and image/speech recognition and the implications should AI surpasses HI in all fields, arriving at General AI (AGI). There is also a consensus that AI is bound to affect all aspects of our lives and societies, with exceptional impact on jobs and income inequality, but disagreement on the dangers that AGI may pose to humanity, as well as on the possibility of exploiting advances in AI and related technologies to augment our own

---

S. Makridakis (✉) · A. Polemitis  
University of Nicosia, Nicosia, Cyprus  
e-mail: [makridakis.s@unic.ac.cy](mailto:makridakis.s@unic.ac.cy)

intelligence. The two authors have been debating for more than two years on whether AI is approaching HI or is closer to that of a hedgehog. They present their common position in this chapter that starts by providing two different definitions of intelligence, showing that accepting one or the other affects the discussions of the nature of AI and its present and future accomplishments. The paper then moves to the first part, comparing such achievements to those of HI, concluding that they are of different, complementary nature, while also presenting its current capabilities and future challenges. The second part discusses the future of AI and the uncertainties and challenges it faces. Finally, part three presents four possible scenarios, including Intelligence Augmentation (IA) and its critical role in substantially improving HI by exploiting progress in AI and advances in the related technologies of nanotechnology and neuroscience. There is also a concluding section deliberating about the long-term future of AI by presenting the views of the two authors: one believes that ultimately, AI will dominate HI and the other argues that breakthroughs in augmenting HI will at least keep up with advances in AI.

## DEFINING INTELLIGENCE

The saying among psychologists is that there are as many definitions of intelligence as there are experts asked to define it (Legg and Hutter 2007). Looking at these definitions, it becomes clear that any comparison of AI to HI will greatly depend on the definition used. The simpler definition given by the American Heritage Dictionary (fourth edition, 2000) defines it as “The capacity to acquire and apply knowledge”. At the other end of the scale, Sternberg (1997) defines intelligence as “comprising the mental abilities necessary for adaptation to, as well as selection and shaping of any environmental context”. Between these two definitions, there are plenty of others, including distinctions of practical, analytical, emotional, and creative intelligence, as well as various definitions provided in AI publications (Alayon 2018). According to the first definition, a computer’s ability to recognize a face can be classified as intelligence, with the added advantage of being able to identify in minutes the face of a person in a stadium filled with thousands of people. According to the second definition however, even AlphaGo Zero could not pass the test of being defined as intelligent, as it can learn but not adapt or shape the environment. The same is true with the superb ability of Autonomous Vehicles (AVs) that “learn” to drive on their own, but are unable to

operate and adapt to “new” situations. Interestingly, it may take decades for AVs to be able to do so as auto executives admit at present (Carey and Liemert 2022).

In all cases, the meaning of learning must be explored further as there are differences in the way humans and machines learn. For instance, machines, or to be more precise algorithms, recognize faces by measuring the unique distances among some 80 nodal points and comparing them against a data bank holding corresponding information of millions of faces in order to identify a specific person. Humans recognize faces using a holistic approach that may involve the firing of thousands of neurons (Panko 2017). Are algorithmic and human face recognition similar? There are some definite similarities and some critical differences. A young child learns to recognize the face of his/her mother without any previous knowledge (but so does AlphaGo Zero that learns by starting from scratch), while also discovering from a young age to interpret facial expressions. There are similarities and differences, as well as advantages and shortcomings, between the algorithmic and human approaches to face recognition, with the two achieving comparable results, but by utilizing what seem like complete different approaches. While acknowledging differences, we cannot tell whether one approach is better (more intelligent, more creative, more resourceful, more efficient) than the other, or if they are based on similar or different principles. Philosophers have begun and will continue to argue about this question for a long time.

In this paper, we start by admitting that HI and AI, in their present form, are two dissimilar but complementary forms of intelligence, with AI surpassing HI in games and image, speech and text recognition, while HI excels in most other tasks involving open systems and situations/decisions where the objectives are not clear, the rules are not known, or the environment can change. The two critical questions are: can AI and HI coexist, each contributing where they excel while avoiding their weaknesses? And how long will it be before AI advances considerably in other areas of applications approaching/exceeding HI and in doing so achieving AGI?

### *The Distinctive Nature of AI and HI*

Humans learn by recognizing patterns in the environment and, unsurprisingly, the same principle also applies to AI that attempts to discover

patterns in large volumes of data, utilizing algorithms such as Machine Learning (ML) as well as its many variations like DL. Recognizing patterns in data is not, however, the exclusive ability of AI, as it is also shared by other fields. Engineers, for instance, strive to separate the noise from the data to come up with the signal or pattern. The same is true in the statistical field of forecasting where the randomness is separated in order to identify the underlying pattern in the data. The big novelty of AI is the use of deep reinforcement learning to achieve breakthrough results. According to David Silver (2018), the lead researcher of AlphaZero, the algorithm starts from scratch, without knowing everything humans do about Go, chess, or shogi, and learns on its own, given only the basic rules and some goals to be achieved. Silver ascertains that as “*AlphaZero has to figure out everything for itself, every single step is a creative leap. Those insights are creative because they weren't given to it by humans. And those leaps continue until it is something that is beyond our abilities and has the potential to amaze us*”. This is definitely a form of brilliant *learning*.

Pearl and Mackenzie (2019) express a different standpoint. They maintain that what AI does now may involve some learning that cannot be applied, however, beyond the restricted areas of games and image, speech, and text recognition. They are claiming, therefore, that a breakthrough is needed before machines and robots can really learn to be intelligent. Their answer is that algorithms have to be capable “*to think rationally about cause and effect, credit and regret, intent and responsibility*”, not just recognize patterns in vast volumes of data, which is something that was already being done well before AI appeared. **The Book of Why** contains their proposals on how to achieve these goals, admitting though, that it would take a very long time to develop and implement an approach where robots and machines would be capable of exhibiting a human type of intelligence by understanding meaning and figuring out causal links. A similar viewpoint is made in a new book by Marcus and Davis (2019) who argue that despite the hype surrounding AI, creating an intelligence that rivals or exceeds human levels is far more complicated than we have been led to believe as AI cannot understand meaning, causal relationships, as well as space and time.

Is it possible to bridge the gap in Silver's approach on the one hand and those of Pearl/Mackenzie and Marcus/Davis on the other of what constitutes intelligence? Conventional thinking may suggest that it may be worth trying, but it might turn out that two dissimilar but complementary approaches to intelligence will be more valuable than

two overlapping ones competing with each other. It would be beneficial, therefore, to accept that at least at present, AI is fundamentally different to HI and attempts to exploit the advantages of each while avoiding their shortcomings.

### *The Evolution of HI and Its Complementarity with AI*

The evolution of HI spans over a period of around seven million years, while that of AI is hardly older than fifty, with the great majority of its major advances having taken place in the last few years. In this respect, AI's achievements are impressive, inspiring its proponents to declare that since AI has progressed so rapidly in less than half a decade, there will be no limits to what it can achieve in the long run as growth accelerates. Kurzweil (Kruger 2018), the most optimistic of its supporters, predicts that AI will reach HI by 2029 and achieve Singularity<sup>1</sup> in less than two decades after that. At the same time, the impact of HI on our planet has been far reaching and its dominance over all forms of life is unquestionable. Equally important, humans have not only managed to adapt to tough environmental changes, but have also succeeded in shaping the environment to their own advantage, achieving absolute supremacy among other animals. The emergence of AI has brought a new entrant to the intelligence race, potentially capable of challenging HI's supremacy or alternatively, further enhancing HI, by accelerating its improvements and contributing to the broader attainment of desired human goals. There is, however, profound disagreement about the future progress of AI and its time path toward AGI, which is covered in the second part of this paper.

If we accept that AI and HI are different, then we need to better understand their advantages and limitations by considering Tables 1.1 and 1.2. Table 1.1 shows available AI applications that have surpassed or are approaching HI. They include games, image, speech, and text recognition, and AVs. AI has two big advantages in games. It can accelerate its learning by playing millions of games in the span of a few days, and it can exploit brute force to apply its learning strategy. For these reasons, winning games is and will remain a unique AI competence that no human can challenge at present or in the foreseeable future. At the same time,

<sup>1</sup> Technological singularity is a hypothetical future point in time at which technological growth becomes uncontrollable and irreversible, resulting in unfathomable changes to human civilization.

games, as their name implies, are not a part of reality. For one thing, the rules governing them are known and do not change over time, and moreover, there is no doubt on what to do to win. In other words, the objective is absolutely transparent and does not change over time. Games are close systems.

Beyond games, the rules of the remaining applications of Table 1.1 are also known and fixed, while several additional conditions shown in Table 1.2 must also be satisfied for AI to be able to surpass or approach HI, with a critical factor being that in real-life applications the noise or uncertainty is much greater than in all the applications listed in Table 1.2. Finally, none of the AI applications in Table 1.1 or 1.2 are capable of *adapting* to environmental changes, or even operating under new conditions, demonstrating HI’s unique abilities and AI’s biggest limitations. In the next section, AI’s current capabilities and the substantial challenges of the applications in Table 1.1 or 1.2 are considered and their implications outlined.

### *AI: Current Capabilities and Future Challenges*

There is no doubt that AI’s achievements are impressive. From the four groups of applications shown in Table 1.1, AVs hold the greatest potential and the biggest challenge for their successful implementation as there is a growing concern among AI experts that it may be years, if not decades, before self-driving vehicles can reliably avoid accidents and be able to operate under *all* driving conditions (Brandom 2018). Over-optimism and rising expectations about future achievements seem to be rampant in the AV industry. In February 2019, Elon Musk promised a truly self-driving Tesla car by the end of 2020, saying “*it will be so capable, you’ll be able to snooze in the driver’s seat while it takes you from your parking lot to wherever you’re going*”. Musk had made a similar prediction in 2016 when he announced that Tesla’s cars would support fully-autonomous cross-country driving by the end of the following year (Brown 2018). Clearly, his 2016 prediction turned out to be wrong, and according to skeptics, his recent one of 2019 will have the same fate. Musk is not the only one “overselling” AVs’ capabilities; so did Google, Delphi, and MobileEye, while NuTonomy has announced plans to deploy thousands of driverless taxis onto the streets of Singapore in 2019 and greatly reduce fares by doing so. Is such optimism realistic?

**Table 1.1** Major available AI applications superior to or approaching HI

<i>Major AI applications Vs. Real-life events</i>	<i>Objective:</i> <i>An exact transparent &amp; precisely defined</i>	<i>An exact algorithm linking input &amp; output can be discovered &amp; used</i>	<i>The environment can change unpredictably</i>	<i>Adaptation possible when environment changes</i>	<i>Amount of noise/ Extent of uncertainty</i>	<i>Examples</i>
Games: Perfect Info	Yes	Yes	No	No	None	Go; Chess; Shogi
Games: Imperfect Info	Yes	Yes	Yes	No	Considerable	Starcraft; Heads-up, No limit Texas Hold'em Poker
Image/speech/text recognition	Yes	Not always but recognition can be achieved using big data	No	No	Moderate to large (can be reduced using big data)	Face recognition, identifying cancer in X-rays, automatic translations, text to voice transcriptions
Personal assistants/recommendation engines	Yes	Not always but recognition can be achieved using big data in the cloud	In a limited way	No	Moderate to great (can be reduced using the cloud)	Alexa, Siri, Google assistant, product recommendation engines

(continued)

Table 1.1 (continued)

<i>Major AI applications Vs. Real-life events</i>	<i>Objective: Transparent &amp; precisely defined</i>	<i>An exact algorithm linking input &amp; output can be discovered &amp; used</i>	<i>The environment can change unpredictably</i>	<i>Adaptation possible when environment changes</i>	<i>Extent of uncertainty</i>	<i>Amount of noise/</i>	<i>Examples</i>
Autonomous vehicles (AVs)	Yes	Great efforts are being made in this direction with mixed results	Yes	No	Large to great (unpredictability can cause accidents)	Driverless cars, trucks, drones, pilotless airplanes	
Robots	Yes	Great efforts are being made with limited success	Yes	No	Large to great	Pepper, Sophia, Boston Dynamics' Atlas	
Real life happenings/ events	No	In a limited number of cases	Yes	Yes	Great: Fat-tails; Black Swans	Deciding what to study, dating, accepting a job offer, making strategic/investment decisions, buying a house, getting married	

**Table 1.2** Various types of AI versus real life applications

<i>Type of Major Application</i>	<i>Major Achievement</i>	<i>Example(s)</i>	<i>Remarks</i>
<b><i>Games</i></b>			
Go, Chess and Shogi	Beat the world's champions	AlphaGoZero	Perfect information
StarCraft	Beat some of the world's best players	AlphaStar	Strategy game with short- and long-term objectives
Heads-up No-Limit Texas Hold 'em	Beat world-class poker players	Libratus	Imperfect information
<b><i>Image recognition</i></b>			
Face Recognition	Achieved accuracy: 95.12–99.63%	FaceNet	Great progress over the last few years with important applications in fields like AVs, medicine, payments, internet applications
Image Classification	2017 accuracy of 97.75%	Imagenet	Great progress over the last few years
Handwriting	2018 accuracy of 97%	MetaMoji, Notes Plus	
<b><i>Speech/Text Recognition</i></b>			
Speech-to text/Text-to-speech	Approaching, and in some categories even surpassing, human-level performance, accessible on smart phones and smart gadgets	Dragon Home 15	Great progress over the last few years
Automatic Text Translation	Approaching and in some categories even surpassing human-level performance, accessible on smart phones and smart gadgets	Apps in mobile phones	Great progress over the last few years with important applications becoming available on smart phones and home devices that provide personal assistance and individualized recommendations

(continued)

**Table 1.2** (continued)

Type of Major Application	Major Achievement	Example(s)	Remarks
Automatic Speech Translation		Apps in mobile phones	
Personal Assistants		Alexa, Siri, Google	
Recommendation Helpers		Google, Netflix	
Smart Speakers		Amazon, Google	
<b>Autonomous Vehicles (AVs)</b>			
Busses	Self-driving following fixed routes providing improved service	Baidu's Apolong	They follow fixed routes, simpler to handle than traveling by car/taxi
Cars	Promising revolutionary improvements when available	Waymo, Tesla	AV technology for all situations/ weather/ situations not yet available
Real life happenings/ events	Master and control the environment on earth	Great scientific achievements	Human Intelligence (HI) accountable for human supremacy over all other forms of life on earth

NYU's Gary Marcus, an AV expert, talks about the need for a serious recalibration of expectations concerning AVs that may be delayed considerably before self-driving cars will be safe to be put onto busy streets, able to operate under all weather conditions, and capable of avoiding fatal and other accidents (Brandom 2018). The shift toward accepting the limitations of AVs is shared by Krafcik, Waymo's CEO, who recently confirmed "*It'll be decades before autonomous cars are widespread on the roads and even then, they won't be able to drive themselves in certain conditions*". According to Krafcik, the problem adding to the uncertainty is: "*You don't know what you don't know until you're actually in there and trying to do things*" (Tibken 2018). Where do these difficulties leave AVs and the promises that they will be fully operational, at the latest, by 2019? Even the leader Waymo has been playing conservatively, by introducing driverless taxi services in some Phoenix Arizona suburbs with large, but

not overcrowded roads, where heavy rain is rare and there is practically no snow, with a safety driver to take over in case of trouble.

The critical question is, how long will it take AV technology to overcome the practical problems? In the pessimistic scenario, AVs may have to follow Tesla's approach and operate driverless most of the time until the system demands that the driver takes control. But such a solution is far from the fully automated AV that will come to pick you up when you call, take you to your destination, and depart to do the same for its next customer(s), reducing transportation costs and traffic congestions considerably, and making owning a car uneconomical. This scenario may be far away into the future, as many practical problems will need to be resolved before AVs can be utilized on a grand scale (Ford 2018, pp. 429–430). An alternative way to Waymo's is that of startups like Voyage, Optimus Ride, and May Mobility, whose strategy is to "simplify the driving world" by operating in retirement communities and other geographically confined, controlled environments with little traffic and low speed limits (Coren 2018). In such settings, the opposite of driving in busy cities, AVs can operate effectively without all the problems of heavy traffic and all weather driving, while providing an indispensable service to, say, retirees who are unable to drive their own car. Perhaps this is a more realistic strategy to full service AVs operating on crowded roads and fast lane highways, thus becoming a testing ground to gain experience with operating AVs in "real-life" environments.

Are AVs' troubles/criticisms valid? Every year there are more than 30,000 fatal accidents in the USA alone, which are not publicized, apart from special cases of pile-ups or unusual crashes. However, there were first page stories when an Uber AV killed a woman, or when a Tesla car accelerated and hit a concrete barrier while on autopilot. Public opinion does not seem to trust AVs. According to a survey released by AAA, 71% of respondents said that they were afraid to ride in a self-driving car (Novak 2019) while residents in an Arizona suburb slashed the tires and pelted rocks at Waymo's AVs. Are they right? Are AVs more dangerous than human drivers? An objective approach would be to compare accidents caused by AVs and humans in an impartial, objective way. If AVs operating in their present state cause considerably fewer accidents than human drivers, then they should be allowed to operate on the streets and highways without a driver. Although there are possible difficulties and unexpected problems with AVs, this does not mean that their introduction should be unduly

delayed until all concerns are resolved. After all, 94% of serious car accidents are caused by human error, whereas about one in three by drunk drivers. If it is accepted that “to err is human”, then it should also be tolerated that AVs can also commit errors and be forgiven as long as their rate of accidents is much lower than that of humans. What is certain is that AVs will not drive under the influence of alcohol or drugs, at least for the foreseeable future, and they will not break traffic regulations or exceed speed limits.

It is argued that AVs should not be put into operation because of their difficulty to drive in snowy conditions, during bad weather, or at night. Is this any different for people living in places where it rarely snows, having great difficulties and many accidents when driving on snowy roads? The same is true for people who have problems seeing and hearing well, or driving at night when the number of accidents is more than double than that of driving during the day. Finally, from a societal standpoint, there must not be any perceived difference between deaths from car accidents driven by a human versus those driven by AVs. The objective in both cases must be that of minimizing the overall number of deaths without discriminating about their origin, although it would admittedly be difficult to achieve such an objective as people are and will continue to perceive AV accidents and deaths much more severely than those caused by human drivers. Some hard decisions will have to be made in the future about AVs and when they will be allowed onto busy city streets and highways, even though they may cause fewer accidents and deaths.

The remaining applications shown in Tables 1.1 and 1.2 are image, speech, and text recognition that include personal assistants and recommendation helpers that are the most developed, widest used, and most promising of all AI applications for a great number of people. These applications are available, not only on all types of computers, but also on smart phones and other gadgets like Alexa’s speakers. Given their widespread usage billions of times every single day, there is a strong competition among the major tech firms providing them to improve them as much as possible and further increase their utilization. Google is the leader in most applications in this category with its search engine, machine translations, Google Assistant, and other services. Its deep learning group’s continuous innovations substantially reduce response time in voice recognition and other services offered by the tech giant (Tung 2019). Amazon is also continuously improving the performance of Alexa that understands the human voice and is used as a personal assistant, in addition to being a

music speaker, and costs less than \$150. There is also a considerable effort being made toward continuous improvements in all kinds of image and face recognition by more than two dozen established firms and numerous startups. Such an effort will continue into the future providing more efficient neural networks and faster GPU processors to further improve these applications. Their implications and challenges are considered next.

## THE FUTURE OF AI: UNCERTAINTIES AND CHALLENGES

Although there is considerable agreement that AI is a disruptive technology with a profound impact on all aspects of our lives, work, and society, there is widespread disagreement on its future path and achievements, including whether or not it will surpass HI during this or the next century. At the same time, all large tech companies are spending billions of dollars a year on AI research and buying AI startups while, according to PwC and CB Insights reports, the VC funding for AI startups increased by a staggering 72% last year reaching a record \$9.3 billion (Chapman 2019). Part of such funding goes to newcomers targeting niche AI markets, in order to be able to compete with the tech giants by identifying and satisfying the particular needs of specialized markets (Kossuth and Seamans 2018). What will all these investments in AI bring? Will they follow the path of AVs that can drive in Phoenix but not Alaska, or will they provide some breakthrough, eventually leading to superintelligence?

These issues are covered in two books, **Architects of Intelligence**, Ford (2018), and **Possible Minds**, Brockman (2019), by 45 leading AI experts, and although they all agree on AI's profound achievements, they disagree on everything concerning its future course and impact. The account of Rodney Brooks (Ford 2018), one of the world's foremost roboticists and co-founder of *iRobot* and *Rethink Robotics*, is revealing. His statement "*We don't have anything anywhere near as good as an insect, so I'm not afraid of superintelligence showing up anytime soon*" (p. 431), from someone who has spent most of his professional life building robots, is an eye opener to the hype surrounding a good part of AI. In the same direction is his response to the statement "*Deep and reinforcement learning is being used to have robots learn to do things by practicing or even just by watching YouTube videos*". His answer is surprising: "*remember they're lab demos*" (p. 432) and then continues by saying that robots are built to perform specific tasks and that the impressive demonstrations

on TV or videos using robots “*were cooked, very, very cooked*” (p. 428). Moreover, his answer to the question on the progress of his work in creating robotic hands was “*that work is progressing slowly. There are various exciting things happening in lab demos, but they’re focusing on one particular task, which is very different from the more general way in which we operate*” (p. 432).

Brooks is on the opposite end of the spectrum to Kurzweil. His answer to the question of what would be “*your best guess for a date when there would be at least a 50 percent probability that artificial general intelligence (or human-level AI) will have been achieved*” was the year 2200 (versus 2047 by Kurzweil). This is 182 years from 2018, and that answer was given with a 50% probability, not with any higher level of certainty. Characteristic of Brooks’ attitude toward AI was his statement that the most successful of his robots has been Roomba, a vacuum cleaning one that has shipped more than 20 million units and only has the brain of an insect.

The views of the majority of the remaining 43 AI experts are between those of Kurzweil and Brooks. Their average guess of when AGI will arrive is at the end of this century, which is also the median of their estimate. In summary, **Architects of Intelligence** and **Possible Minds** provide a wealth of fascinating information about all aspects of AI. At the same time, however, there is a clear lack of agreement among the very people who have created AI and have spent their whole life researching or developing successful AI applications. The same disagreement prevails about the social implications of AI on unemployment, quality of jobs, and income inequalities.

### *AVs: Achievements, Limitations, and Future Prospects*

AVs present an exceptional application exemplifying AI’s impressive achievement. Having a car capable of driving on its own on a highway or a busy street is a phenomenal success, unimaginable just a few decades ago, even if there are still mishaps and occasional accidents, some of which are fatal. At the same time, there is little doubt regarding Marcus’ comment about the need for a serious recalibration of expectations concerning AVs that may be delayed considerably before they are safe enough to be put onto busy streets (see also Boudette 2019). Even if AVs can operate safely 99.9% of the time, this is not enough, as the remaining 0.1% can create serious problems, in particular as the perception of fatal accidents by AVs is much more severe than similar ones caused by human drivers. In a super

rational society, the only important concern should have been a comparison of the number of accidents by AVs versus those of human drivers, to decide when AVs should be introduced. Since, however, decisions will be influenced by human perceptions, the introduction of AVs on the streets will take much longer, even if doing so may reduce the overall number of accidents. This means that AV companies will be extremely careful to avoid fatal accidents, which would cost them not only huge settlement compensations but, in addition, a great deal of negative publicity. Therefore, they are obliged to be remarkably conservative and take as few risks as possible and as a result slowing progress.

### *The Current Situation*

In 2018, in California where reporting was obligatory, there were thirteen companies testing AVs on its roads. From those, Waymo, an Alphabet company, ran 1.27 million miles on the road using AVs, more than twice the total of all other twelve firms. During those miles, its cars were “disengaged” (that is the driver took over control of the car) once every 11,017 miles (the other companies had higher disengagement rates) and only 24 minor accidents ([Vardhman 2019](#)). What do these statistics tell us? Unfortunately, no direct comparisons can be made with human driving as the two situations are not comparable. What is certain is that AVs can drive safely for millions of miles around the USA, while 40% of deaths in the USA are caused by drunk drivers and another almost 30% from speeding, which would be eliminated by AVs. In addition, we know that the performance of AVs is improving over time (for instance the disengagement rate of Waymo’s cars dropped to about half in 2018 from 2017). At the same time, we know that AVs’ software seems to have problems dealing with pedestrians while they are also causing more traffic jams than human-driven vehicles. Worse, there is the chance of something new and unexpected to occur that the software cannot identify or be able to deal with. If there is a test driver, he/she can take control, but what if there is no one? For these reasons, there is a growing pessimism that AVs will be able to drive without a driver in the foreseeable future ([Carey and Liemert \(2022\)](#)).

## THE FOUR AI/HI SCENARIOS

Given the disagreement about the future of AI and being aware of our failures to predict long-term technological progress, our approach is to accept the impossibility of providing any specific predictions and instead present four possible scenarios and their implications, hoping to cover the entire range of what could happen in the future.

### *Smarter, Faster Hedgehogs*

This is a likely scenario with constant marginal improvements in the DL and other AI algorithms. Its implications could be small, but with continuous improvements providing smarter and faster ways of applying AI to practical problems. Such improvements coupled with faster computers and GPUs can accelerate the value of hedgehogs and make them more valuable. AVs can be improved in a variety of ways, for instance by using more efficient algorithms, better sensors and LIDARS, or better recognition of when the driver must take control of the vehicle.

### *An Army of Specialized Hedgehogs*

This would mean that there will be AlphaZero to play Go, Chess, and Shogi; AlphaStar to play StarCraft; AlphaFold to predict the 3D structure of protein, and other specialized algorithms to play different games or excel in specific tasks. The same principle may apply to image recognition with one algorithm learning to recognize cancers, another to identify chemical compounds, and another still to analyze photos in order to identify potentially habitable exoplanets. There are no limits to how many such algorithms can be developed, each one specializing in some specific area, providing huge benefits first in the type of applications shown in Tables 1.1 and 1.2 and then in other specialized areas, including medicine, science, the Internet of Things, and management.

Although it is intellectually challenging for AlphaZero to be able to play three games, it is a step toward AGI. It would make no practical difference if there were three different programs: AlphaGo, AlphaChess, and AlphaShogi, instead of just one, AlphaZero. As long as the specialized AI programs can achieve the tasks they were intended to perform, as well or better than humans, there is no difference whether this is done

by a single program or three. Perhaps the importance of AGI is exaggerated, as the majority of real-life tasks can be dealt with equally well with narrow AI. A face recognition program, for example, does not need to understand natural languages, while Alexa does not need to recognize faces, at least at present. We may need to accept that narrow AI is the equivalent of the specialization trend that is continuously growing, as knowledge is advancing exponentially in our modern societies. There are clear advantages, for instance, to going to a specialized doctor in case of health problems rather than to a generalist. This does not mean that the search for AGI should stop with certain applications such as robots, or that a higher level of intelligence than that of a hedgehog may not improve its performance and usefulness. However, if we acknowledge that AI is completely different and complementary to HI, then practically we may be better off if we exploit its present advantages as much as possible, leaving the intellectual pursuit for AGI for later.

The economic and social implications, should this scenario prevail, will be a continuation of recent AI trends, meaning that some jobs will be lost and new ones will be created, while wealth inequality continues to increase. The envisioned changes according to this scenario would probably accelerate what is already being done and may reach the level of disruption, depending on the cumulative effects of the influence of the numerous super-specialized hedgehogs.

### *Fully Integrated HI and AI*

This scenario is substantially different from the first two, advocating a different future where humans exploit advances in AI and related technologies to amplify their own intelligence. Such a scenario is based on two premises. First, humans have shown an extraordinary ability to adapt under severe environmental shifts; therefore, it is highly unlikely for them to stay still in the face of the greatest possible existential threat they may face by an approaching AGI. Their obvious defense will have to be to augment their own Intelligence to keep up with advances in AGI. Such Intelligence Augmentation (IA) can be achieved by brain-enhancing technologies that would exploit advances in AI to improve human capabilities. *“Instead of just fretting about how robots and AI will eliminate jobs, we should explore new ways for humans and machines to collaborate”*, says Daniela Rus (2017), the director of MIT’s Computer Science and Artificial Intelligence Lab. Second, it seems that suitable technologies, now in

their infancy, are being explored by startups aimed at amplifying human intelligence.

By harnessing the speed and memory of computers and the possibilities provided by AI, there is nothing to prevent people from being able to enhance their ability in playing games, even if eventually they cannot beat AI programs that at present maintain a clear advantage over humans. The centaur idea is supported by Moravec's Paradox, which states that "it is comparatively easy to make computers exhibit adult-level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility". Hence, there is a perfect complementarity that can be explored beyond games to areas such as strategy, where humans excel, and tactics, where computers and AI possess a great advantage. IA can naturally amplify human intelligence by exploiting AI's considerable brute force and people's higher-level intellectual abilities. In a perfect future, there could be a functioning symbiosis of humans and machines rather than an adversarial face-off in a lose/lose competition. Such symbiosis will reduce the perceived danger that one day AI will end human supremacy and keep HI at least on a par with AI.

### *More Efficient Ways of Communicating with Computers*

It may seem like prehistory, but it was hardly 50 years ago when the only way to communicate with computers was with "punch" cards for both inputting programs and data into computers and receiving the output. The procedure involved was time-consuming and tedious as the cards first had to be punched and then submitted to a computer operator to run the program on a mainframe, batch computer. Consequently, the computer punched the output of the program onto cards that had to be given to the user, to put them into a printer to print the output of the program. In case of even a simple mistake, like omitting a comma or misspelling a command, the computer rejected the submitted program and the user had to correct the mistake and submit it again.

An amazing improvement, if not a revolution, was keyboards directly connected to a computer to enter the program/data and screens to see the instructions and output, which greatly improved computer efficiency by allowing direct interaction with the computer. The next equally significant improvement will be to make both keyboards and screens obsolete. As computers are approaching humans in natural language processing,

keyboards could be replaced by voice commands, where typing documents or emails would be done using dictation. The output of the computer could also be provided in computer speech or be projected onto any surface instead of a computer screen. Computer usage without keyboards/screens would considerably improve receiving or processing information and could be done with the cloud, using a smart phone or even a digital watch. All desired information would therefore be readily available at any time and place leading to the first phase toward using IA to augment HI.

### *Direct Brain to Computer Interface (BCI)*

The second phase to IA has already begun (Etherington 2019) and involves progressing from voice communications with the computer to direct links between the mind and computers. This kind of direct brain-to-computer communication can be of two types: non-invasive and invasive (Gonfalonieri, 2018; Clarke, 2019). The first can be achieved through some external device that can transmit brain signals to the computer directly or through a software program (a predecessor to BCI was late Steven Hawkins' Intel constructed computer that could read movements of his cheek muscle and translate them into speech). The second type would require an invasive approach where a tiny chip would be implanted in the cortex region of the brain. This is already done, allowing paraplegics to move their paralyzed limbs, or use a computer. At present in addition to Neuralink (<https://www.neuralink.com/>), there are a number of startups like Kernel (<https://kernel.co/>) and Nanalyze (<http://www.nanalyze.com/>) developing applications for both types of BCI with science fiction implications, if successful.

### *Non-Invasive Forms of Interfaces*

This type of BCI approach can be achieved using a helmet (Hall 2018), a headband (Scott 2014), or some software application (Brodin 2018). At present, they are mainly used by people with spinal cord or motor problems to be able to use their hands or feet, while future applications envision expanding the range of applications and achieving IA capabilities through direct brain-to-computer information processing and retrieval. This would be the equivalent of being able to interrogate Google directly in one's head and processing the information obtained to make decisions. Companies like Kernel work exclusively to develop this type of non-invasive technology, maybe not for tomorrow, but further into the

future, initially helping those who are paralyzed and later extending the technology to healthy people.

### *Invasive Forms of Interfaces*

Invasive forms are similar to non-invasive ones except they require a surgical procedure to insert a chip, or even a tiny CPU like the one constructed by IBM (Jones and Wagner 2018), in the cortex region of the brain. Given the possible danger to the brain, this type of BCI is only done in case of serious paraplegic problems and when no other alternatives are available. However, there are many cases where serious paraplegics have been able to move their limbs through this kind of intervention. Companies like Elon Musk's Neuralink aim to exploit this technology and offer a needed service not only to paraplegics, but also to the healthy wishing to communicate more efficiently with computers and augment their intelligence via such a direct link (Urban 2017). Invasive forms of BCI encompass some serious dangers if something goes wrong with the operation, creating problems with the function of the brain. Therefore, it will be a long time before permission to use them on healthy people is granted. However, there are new advances in science and technology opening up brand new possibilities not only in BCI, but also in Brain-to-Brain Interfaces (BBI) where communication can be achieved exploiting brain waves.

### *Brain-To-Brain Interfaces (BBI)*

Attempts to achieve BBI go back to 2014 (Grau et al. 2014) but have accelerated in the last couple of years. BrainNet is a multi-person Brain-to-Brain Interface (BBI), developed at Cornell University, allowing direct collaboration between brains (Miley 2019). In experiments, two subjects act as “Senders” whose brain signals are read in real time using EEG data analysis to extract their decision about whether to rotate a block in a Tetris-like game before it is dropped to fill a line. Consequently, this information is sent via the internet to the brain of the third subject, the “Receiver”, who cannot see the game screen and makes a decision using an EEG interface about whether to turn the block or keep it in position. In tests, five groups of three subjects successfully used BrainNet to perform the Tetris task, with an average accuracy of a little more than 81%.

In another application, scientists at the University of California, San Francisco, developed a brain implant that can read people's minds and

turn their thoughts into speech (Gallagher 2019). The technology works in two stages. First, an electrode is implanted into the brain to pick up the electrical signals that “maneuver” the lips, tongue, voice box, and jaw. Then, powerful computing is used to simulate how the movements in the mouth and throat would form different sounds, resulting in synthesized speech coming out of a “virtual vocal tract”. According to Professor Edward Chang, one of the researchers of this study, published in the prestigious Nature (Anumanchipalli et al., 2019), “For the first time, our study demonstrates that we can generate entire spoken sentences based on an individual’s brain activity”. Such studies identifying and controlling brain waves are a first step toward a wider utilization of BBIs and we will see more of these types of applications in the future, as more sophisticated brain scans are combined with artificial intelligence to produce tools that can detect and appraise brain waves.

### *The Future World of Transhumanism*

Technologies rarely develop separately of each other, as inventors aim to exploit the advantages of each to expand the range of their applications. Computer speed and memory could not have followed Moore’s law without momentous progress in nanotechnology that permitted the shrinking of chips, making them cheaper and more powerful, which in turn allowed the building of superfast computers capable of efficiently running DL algorithms and bringing progress in AI. Similarly, advances in neuroscience permitted BCI that could allow IA to become the objective of the startup Kernel, mentioned above, whose mission is “to build a non-invasive mind/body/machine interface (MBMI) to radically improve and expand human cognition”. The ultimate conclusion to all of these technologies combined leads to transhumanism (Benedikter and Fathi 2019), the possibility that the human race can expand its current mental, physical, and emotional abilities significantly, and by doing so overcome its biological and mental limitations, through advances in neuroscience/nanotechnology and AI, that would include BCI, Brain-to-Brain communications (BBI), or MBMI. These possible developments could have been considered an additional scenario to the six presented above, but they seem too speculative and far-fetched at present to be considered as such. At the same time, some bright people seem to believe in them as they are spending billions of dollars searching for commercial applications of these technologies that could create the new Google or Apple.

What will become reality and what will remain in the domain of science fiction? We will leave this question for the reader to answer—keeping in mind that technology can exceed science fiction in a period of less than 200 years. A characteristic example is that of Jean-Baptiste Say, the famous French economist who in 1828 predicted that it would be impossible for cars to replace horses in the middle of big, busy cities (Say [1828](#)). Could Say have ever believed that there would be self-driving cars in the future? Or could Watson, IBM’s president who foresaw in 1943 a maximum demand of seven computers, that there would be more than seven billion of them less than 80 years later?

## RUNAWAY AGI

The big wager and great unknown is if and when a breakthrough will occur that would allow AI to understand natural languages, be able to reason, acquire common sense, be capable of making causal inferences, and in general be able to attain traits similar to those of human intelligence. At present, we are not sure whether such breakthroughs are possible, or when they could be achieved, but the possibility that they could occur sometime in the future is real. After all, it was impossible for people 200 years ago to have believed that cars and airplanes would exist everywhere, while it was not even within the realm of science fiction that cars would be capable of driving on their own. Thus, hard to believe AI breakthroughs cannot be excluded. The interesting question is when they would occur, as top AI experts are not able to agree on a date. From past history, we know that many of our predictions have seriously underestimating technological progress, particularly in the long run, but also predicting technologies that never happened like underwater cities. The challenge is, therefore, to not get too influenced by the hype surrounding future AI achievements, while at the same time, not be too conservative in underestimating future technological progress. Some of the latest AI innovations point toward this direction. **AI paintings** that win prizes, **DALL E 2** making realistic images from a description in natural language, **OpenAI’s GPT** writing screenplays but also poems, composing marketing emails and developing video games, **Google’s** breakthrough conversational **LaMDA** bot, or **GitHub’s Copilot** helping programmers working faster by automatically finishing their code snippets.

Finally, the recently (November 30, 2022) introduced AI **ChatGPT** bot by **OpenAI** is the equivalent of iPhone to the AI field (Savov [2022](#)).

Its ability to interact in many natural languages (English, Spanish, French, Japanese, and Greek among others) is impressive while its answers are remarkable. In a matter of just five days, it attracted a million users impressed by its abilities to answer all their questions in a precise and accurate manner and fascinated by how an AI chatbot can be capable of such advanced abilities to both understand questions in natural languages and having the knowledge to answer them precisely like an expert or a teacher. Clearly, the current version of ChatGPT will be vastly improved when the feedback from the millions of people using it will be incorporated in future versions and as competition between tech giants will intensify (Warren 2023). Clearly, the implications for employment, education, the quality of jobs, and wealth inequality will be affected as people will have to compete with future versions of advance ChatGPT will become available and widely used.

## CONCLUSIONS

For the first time in human history, there are two types of intelligence: born and made; not to use the term “artificial” that could be interpreted as “false” or “fake”. The two authors in their debate agree that these are two different types of intelligence that on the surface appear to have the same purpose, but deep down use different approaches. One of the authors mentions the example of birds and airplanes, saying that both can fly, but their abilities are non-comparable as airplanes can carry huge loads and travel great distances without stopping, while birds are constrained by their biological limits. Adding that no matter how much birds improve, they will never be able to compete with airplanes. The other author believes that the comparison is valid but somewhat unfair as there are no limits to mental capabilities that through IA humans can develop to keep up with made intelligence.

We can believe, one author says, that our brain (computation) is special, unique and inscrutable, while computers (where we can read the code) are obviously just an “algorithm” or “data”. When, through technology, computers reverse this on us and start “reading the code” in our own brains and say “yup, you are likely to say/do this next”, then we will need to rethink “who and what we are”. According to him, there will be no limits to what technology can and will achieve. It might take decades, it might need new breakthroughs like ChatGPT, but in his opinion, it is inevitable it will happen. The history of human scientific progress is to

make the inscrutable scrutable and to take us off our pedestals. Earth is not the center of the universe, it rotates around the sun, actually our solar system is small, actually our galaxy is small, it is actually a grain of sand in the universe. We are the only ones we know who can think and feel the world around us. We are the “brainiacs” on planet earth, but computers are slowly eating away at that. We used to say “they will never beat us at something as complex and strategic as chess” and when this happened, it turned into “well, chess is just a game with fixed rules, of course made intelligence can master it”. These arguments are not based on specific events or expert predictions, but rather on the overall trend of technological progress inferring that it is only a matter of time before “made” intelligence will surpass the “born” one.

The other author accepts that the earth-centric view of the universe is passé and that AI, still in its infancy, is bound to become a formidable competitor to HI in the future. He believes, however, in human ingenuity that will find ways to fend the attack and prevail in the end. It is not the first time that humans have faced existential threats. The cave men/women survived for over ten million years in a hostile environment by developing effective weapons to protect themselves and assure an adequate supply of food from their daily hunting, for themselves and their families. An equal serious threat was overcome during the middle ages when the Catholic Church was using the inquisition to silence any scientific attempt to advance knowledge and open the door to modernity. The author considers these threats equally as serious as the potential one of AI and that humans will find effective ways to defend themselves as they have done many times in their long history. He is certain that the answer will be related to IA, both non-invasive and invasive, that will greatly enhance HI to become capable of staying at least a step ahead of made intelligence. Just as AI is in its infancy, so is IA and its potential to enhance HI. In his recent book, *Superminds: The Surprising Power of People and Computers Thinking Together*, Malone (2018) explores, as the subtitle of the book suggests, how people and computers can help create more intelligent “superminds” through artificial intelligence and hyperconnectivity, that is connecting humans to one another at colossal scales and in rich new ways to exploit their intellect. There are no doubt huge, unexplored potentials in both AI and IA, with any prediction of their future path and implications being highly uncertain, and which view will prevail is unknown. Only time will tell, but we may have to wait a while before we find out. Hopefully, we will not settle the issue, as in science fiction stories, with a war between humans and robots to determine who will prevail in the battle of the “born” versus “made” intelligence!

## REFERENCES

- Alayon, D. (2018). 2045 by Dmitry Itskov, *Future Today*, Medium.Com.
- American Heritage Dictionary. (2000). <https://ahdictionary.com/>.
- Anumanchipalli, G. K., Chartier, J., & Chang E. F. (2019). Speech Synthesis from Neural Decoding of Spoken Sentence. *Nature*, 568, 493–498. <https://doi.org/10.1038/s41586-019-1119-1>.
- Benedikter, R. & Fathi, K. (2019). The Future of the Human Mind: Techno-Anthropological Hybridization?, *ResearchGate.Net*. [https://www.researchgate.net/publication/331983160\\_The\\_Future\\_of\\_the\\_Human\\_Mind\\_Techno-Anthropological\\_Hybridization](https://www.researchgate.net/publication/331983160_The_Future_of_the_Human_Mind_Techno-Anthropological_Hybridization).
- Boudette, N. (2019). Despite High Hopes, Self-Driving Cars Are ‘Way in the Future’, *New York Times*, June 19. <https://www.nytimes.com/2019/07/17/business/self-driving-autonomous-cars.html?smid=nytcore-ios-share>.
- Brandom, R. (2018). Self-Driving Cars Are Headed Toward An AI Roadblock: Skeptics Say Full Autonomy Could Be Farther Away Than the Industry Admits, *The Verge*. <https://www.theverge.com/2018/7/3/17530232/self-driving-ai-winter-full-autonomy-waymo-tesla-uber>.
- Brockman, J. (2019). *Possible Minds: Twenty-Five Ways of Looking at AI*, Penguin Press, New York.
- Brodin, E. (2018). You Can Control This New Software with Your Brain, and It Should Make Elon Musk and Mark Zuckerberg Nervous, *Business Insider*. <https://www.businessinsider.com/brain-computer-interface-startup-nuro-could-beat-facebook-elon-musk-2018-4>.
- Brown, M. (2018). Why So Many Experts Over-Promised on the Arrival of Autonomous Driving, *Inverse*. <https://www.inverse.com/article/49763-how-elon-musk-waymo-and-uber-all-over-promised-on-autonomous-driving>.
- Carey, N. & Liemert, P. (2022). Truly Autonomous Cars May Be Impossible Without Helpful Human Touch, *Reuters*, September 19.
- Chapman, L. (2019). VCs Plowed a Record \$9.3 Billion Into AI Startups Last Year, *Bloomberg*. <https://www.bloomberg.com/news/articles/2019-01-08/vcs-plowed-a-record-9-3-billion-into-ai-startups-last-year>
- Clarke, L. (2019). We Spoke to Neuroscientists About the Future of Brain-Computer Interfaces: How Long Before Human Minds Merge with Machines?, *TechWorld*. <https://www.techworld.com/tech-innovation/we-spoke-some-neuroscientists-about-computer-brain-interfaces-3691918/>.
- Coren, M. J. (2018). The Fast Lane for Self-Driving Cars Runs Through 35 Mph Retirement Communities, *Quartz*. <https://qz.com/1309572/retirement-communities-are-pioneering-self-driving-cars-in-the-us/>.
- Etherington, D. (2019). Elon Musk’s Neuralink Looks to Begin Outfitting Human Brains with Faster Input and Output Starting Next Year, *TechCrunch*. <https://techcrunch.com/2019/07/16/elon-musks-neuralink-looks-to->

- begin-outfitting-human-brains-with-faster-input-and-output-starting-next-year/.
- Ford, M. (2018). *Architects of Intelligence: The Truth About AI from the People Building It*, Packt Publishing, Birmingham, UK.
- Gallagher, J. (2019). ‘Exhilarating’ Implant Turns Thoughts to Speech, *BBC.Com*. <https://www.bbc.com/news/health-48037592>.
- GarmaOnHealthrth*. (2015). Are Ray Kurzweil’s Three Bridges to Immortality Insane? <https://www.garmaonhealth.com/ray-kurzweils-three-bridges-immortality/>.
- Gonfaloni, A. (2018). A Beginner’s Guide to Brain-Computer Interface and Convolutional Neural Networks, *Towards Data Science*. <https://towardsdatascience.com/a-beginners-guide-to-brain-computer-interface-and-convolutional-neural-networks-9f35bd4af948>.
- Grau, C., et al. (2014). Conscious Brain-to-Brain Communication in Humans Using Non-Invasive Technologies, *PLOS*.
- Hall, H. (2018). Ultrasound Helmet Would Make Live Images, Brain-Machine Interface Possible, *Vanderbilt News*. <https://news.vanderbilt.edu/2018/05/08/ultrasound-helmet-would-make-live-images-brain-machine-interface-possible/>.
- Jones, B. & Wagner, J. (2018). IBM’s Blockchain-Ready CPU Is Smaller Than a Grain of Salt, Costs Just 10 Cents, *Digital Trends*. <https://www.digitaltrends.com/computing/ibm-blockchain-computer-salt/>.
- Kossuth, J. & Seamans, R. (2018). The Competitive Landscape of AI Startups, *Harvard Business Review*. <https://hbr.org/2018/12/the-competitive-landscape-of-ai-startups>.
- Kruger, E. (2018). Ray Kurzweil Predicts Machines Will Reach Human-like Intelligence by 2029, *Tech Maven*. <https://sci-techmaven.io/superposition/tech/ray-kurzweil-predicts-machines-will-reach-human-like-intelligence-by-2029-rbm2y0xcT0aqwUhC7wAHiA/>.
- Legg, S. & Hutter, M. (2007). A Collection of Definitions of Intelligence, *Frontiers in Artificial Intelligence and Applications*, 157, 17–24.
- Makridakis, S. (2017). The Forthcoming Artificial Intelligence (AI) Revolution: Its Impact on Society and Firms, *Futures*, June 2017, pp. 46–60. <http://dx.doi.org/10.1016/j.futures.2017.03.006>.
- Malone, T. (2018). *Superminds: The Surprising Power of People and Computers Thinking Together*, Little, Brown and Company, New York.
- Marcus, G. & Davis, E. (2019). *Rebooting AI: Building Artificial Intelligence We Can Trust*, Pantheon Books, New York.
- Miley, J. (2019). BrainNet is The World’s First Non-Invasive Brain-to-Brain Interface, *InterestingEngineering.Com*. <https://interestingengineering.com/brainnet-is-the-worlds-first-non-invasive-brain-to-brain-interface>.

- Novak, M. (2019). 71 Percent of Americans Still Don't Trust Autonomous Cars According to New Survey, *Gizmodo*. <https://gizmodo.com/71-percent-of-americans-still-dont-trust-autonomous-car-1833284527>.
- Panko, B. (2017). How Your Brain Recognizes All Those Faces, *Smithsonian.Com*. <https://www.smithsonianmag.com/science-nature/how-does-your-brain-recognize-faces-180963583/#c51afOdJI93eAikk.99>.
- Pearl, J. & Mackenzie, D. (2019). *The Book of Why: The New Science of Cause and Effect*, Penguin, London, UK.
- Rus, D. (2017). More Evidence That Humans and Machines Are Better When They Team Up, *MIT Technology Review*. <https://www.technologyreview.com/s/609331/more-evidence-that-humans-and-machines-are-better-when-they-team-up/>.
- Savov, V. (2022). ChatGPT Could Be AI's iPhone Moment. Bloomberg, New York, December 12. <https://www.bloomberg.com/news/newsletters/2022-12-12/chatgpt-the-gpt-3-chatbot-from-openai-microsoft-is-tech-magic>
- Say, J. B. (1828). *Cours complet d'économie politique*, Chez Rapilly, Paris.
- Scott, C. (2014). Muse Headband Opens the Door to Brain-to-Computer Applications, *SingularityHub*. <https://singularityhub.com/2014/06/04/heaband-opens-the-door-to-brain-to-computer-applications/>.
- Silver, D., et al. (2018). A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go Through Self-Play, *Science*. <https://science.sciencemag.org/content/362/6419/1140.full>.
- Sternberg, R. J. (1997). The Concept of Intelligence and Its Role in Lifelong Learning and Success, *American Psychologist*, 52(10), 1030–1037.
- Tibken, S. (2018). Waymo CEO: Autonomous Cars Won't Ever Be Able to Drive in All Conditions: And it Will Be Decades Before Self-Driving Cars Are All Over the Roads, John Krafcik says, *CNET*. <https://www.cnet.com/news/alphabet-google-waymo-ceo-john-krafcik-autonomous-cars-wont-ever-be-able-to-drive-in-all-conditions/>.
- Tung, L. (2019). Move over Siri Alexa: Google's Offline Voice Recognition Breakthrough Cuts Response Lag, *ZDNet*. <https://www.zdnet.com/article/move-over-siri-alexa-googles-offline-voice-recognition-breakthrough-cuts-response-lag/>.
- Urban, T. (2017). Neuralink and The Brain's Magical Future, *WaitButWhy.Com*. <https://waitbutwhy.com/2017/04/neuralink.html>.
- Vardnman, R. (2019). 29 Self-Driving Car Statistics & Facts, *Carsurance*. <https://carsurance.net/blog/self-driving-car-statistics/>.
- Warren, T. (2023). Microsoft and Google Are About to Open an AI Battle, *The Verge*, February 7. <https://www.theverge.com/2023/2/7/23587767/microsoft-google-open-ai-battle-search-bing>.



## CHAPTER 2

---

# Expecting the Future: How AI's Potential Performance Will Shape Current Behavior

*James D. Miller*

## INTRODUCTION

Expectations about the future can significantly impact present behavior as individuals make decisions and take actions based on their beliefs about what the future may hold. For example, a twenty-year-old student may choose to invest in a college education with the expectation that it will lead to a higher income over the course of a fifty-year career. Similarly, a thirty-year-old might forgo current consumption to save money for retirement with the assumption that financial stability will continue sufficiently into the future that putting this money in financial markets today

---

By James D. Miller with help from ChatGPT.

I'm grateful to Alexander Felton Miller and an anonymous reviewer for comments on a draft of this chapter.

---

J. D. Miller (✉)  
Smith College, Northampton, MA, USA  
e-mail: [Jdmiller@Smith.edu](mailto:Jdmiller@Smith.edu)

will allow her to enjoy a higher level of consumption forty years hence. However, these expectations can be challenged by the possibility of technological advances disrupting traditional economic and social systems. This raises important questions about how expectations about the future may influence behavior and decision-making in the present.

What if, for example, the twenty-year-old comes to believe that whatever skills he learns in college probably won't be valued by the labor markets in the near future because artificial intelligence (AI) will be able to cheaply do the jobs he is considering training for? Will this twenty-year-old still be willing to devote time and money to his education? Even more radically, what if the thirty-year-old thinks that AI will radically change society so that money saved today won't have value in the future, perhaps because humanity goes extinct, humanity becomes so incredibly rich that any additional money saved today would only trivially increase a saver's future consumption, or that AI will so radically change the world that before his retirement money will no longer have value?

Recent advancements in AI technology, such as OpenAI's ChatGPT, have generated optimism about the potential for further advancements in AI. However, this optimism is tempered by the understanding that these advancements may also have negative consequences for certain industries and jobs. As AI systems continue to improve in their ability to perform specific, narrow tasks, it is increasingly likely that they will outcompete humans in additional fields. For example, AI programs have recently become highly proficient in generating images based on prompts, raising concerns about the future of careers in visual art. Furthermore, as Oxford professor and computational neuroscientist Anders Sandberg wrote "Think about the current uproar about AI art, and now apply the same disruption to a totally unpredictable half dozen human fields every year. That is discounting the hype, which is going to make the actual sense-making and decision making even harder" (Sandberg 2022).

Beyond obsoleting a few occupations, AI holds the potential to radically change human existence. This author has subjectively found that ChatGPT can achieve a grade of A-minus when answering short essay questions for his introductory microeconomics class. While this level of intellectual ability may not necessarily seem groundbreaking on its own, it is important to consider the continual exponential improvement in computing power that can be bought per dollar. Given this trend, it is reasonable to suppose that in as little as twenty or even ten years, AI

systems like ChatGPT may be capable of out-thinking top human scientists and driving significant advancements in technology perhaps creating what has been called a technological singularity, where society changes beyond our current capacity to understand. How would expecting such radical changes in the future influence peoples' behavior today?

## THE BLACK BOX OF MACHINE LEARNING

One of the challenges of predicting the future development of AI is that machine learning, a key component of AI, can often be a “black box” (Doshi-Velez and Kim 2017). This means that it is difficult to understand exactly how a machine learning system reaches a particular decision or prediction (Lipton 2016). This lack of transparency can make it challenging to anticipate the capabilities and limitations of AI systems, as well as the potential impacts they may have on society. In addition, because machine learning algorithms constantly adapt and improve based on new data, it can be difficult to accurately predict how they will evolve over time. These factors make it challenging to accurately forecast the future development of AI.

Because of being developed through machine learning, multiple potential issues might prevent an advanced AI from behaving as its creator's desire. These include reward hacking where “*the objective function that the designer writes down admits of some clever “easy” solution that formally maximizes it but perverts the spirit of the designer's intent (i.e. the objective function can be ‘gamed’)*” (Amodei et al. 2016, p. 2), and mesa-optimization under which the machine learning program creates a separate optimization program perhaps with a meaningfully different objective from the original one (Hubinger et al. 2019).

This lack of transparency and controllability (Yampolskiy 2020) should increase our variance of expectations about what AI may be capable of in the future. As such, it is difficult to simply extrapolate from the current state of AI and make predictions about its future capabilities. Instead, it is akin to a ship sailing uncharted waters in the fifteenth century, where the potential outcomes are vast and largely unknown: You could discover rich new lands, find plentiful supplies of gold and silver (which inflate your economy destroying savings), bring back diseases that devastate and potentially even wipe out your society, or make yourself known to more advanced civilizations that end up conquering you.

## DECREASED EFFORTS

We should be concerned that human efforts may be reduced in various domains as people may expect AI to be able to perform tasks better and faster than humans can. One example of this is in the field of academic writing, as pointed out by AI safety theorist Roman Yampolskiy who posted on Facebook “Feel a bit discouraged from writing. A new paper could take months or even years. Realizing that GPT\* can/will be able to write something good enough in seconds weighs heavily on my future plans” (Yampolskiy 2022). (GPT\* refers to future versions of AIs such as ChatGPT.) Spending, say, four hundred hours writing a paper that in a few years could be generated in seconds by an AI seems like an unrewarding expenditure of time for anyone not up for promotion.

Content producers create movies, books, music, software, and television shows often because they hope to receive revenue from selling this content over decades. But what’s going to happen when people think that soon AI will be better at generating content than humans are and will in a few years flood the market with extremely cheap but high-quality content? We will likely see vastly fewer resources going into creating content when people come to expect AI to be soon good enough to create competing content itself. Under such conditions will young people chose to be educated to fill these jobs?

Real estate development as well might see a slowdown when people begin to expect that in the near future AI-controlled robots will be able to cheaply create desired structures. Indeed, expectations of very productive AIs would inhibit many types of long-term investment where you must spend significant resources today to create something that will survive long-enough to compete with whatever AIs are capable of building in the medium term (Miller 2012, pp. 181–182).

## EDUCATIONAL CHOICES

As AI continues to advance, some students may be concerned that their skills will become obsolete and that they will be unable to compete in the job market of the future (Miller 2012, pp. 191–192). This fear may cause some students to be less likely to spend a significant amount of time and money on a college education. Currently, the US makes aspiring medical doctors go into great debt and spend an enormous amount of time studying dense material because of the (until recently) realistic hope

that if they succeed as a medical student and then as a medical resident, they'll have a lucrative, meaningful, and high-status career as a doctor. But what happens when twenty-one-year-olds estimate that there is a reasonable chance that most doctors will be made obsolete by AI in the next twenty years? Will the US be able to motivate these young adults to devote the rest of their twenties to medical study? If the answer is no, the US is setting itself up for a possible scenario in which it does not train many doctors because it thinks AI will soon be good enough to replace human doctors, but then it experiences another AI winter and becomes stuck with no AI doctors and too few human ones.

Throughout history, technological advancements have brought about both job destruction and job creation. While there is a possibility that the development of AI could result in a more dramatic shift in the labor market than previous technological changes, there is also the potential for AI to create new opportunities for human employment. Even though it is difficult to anticipate what these jobs might be, it is important to remain open to the possibility that new industries and job categories could emerge because of AI development.

Certain types of human labor may be more difficult for AI to automate than others. For example, jobs that involve high dexterity physical labor may be less susceptible to automation than jobs that rely on cognitive skills alone. This may lead to a shift in the types of jobs that are in demand, with more emphasis being placed on skills related to physical labor. As a result, it is possible that ambitious and smart young people may begin to gravitate toward occupations such as plumbing, which could promise to offer high paid employment for longer periods of time than other professions.

Computer programming could be considered one of the jobs most at risk of being taken over by AI. This is because programming involves a high degree of logical thinking and problem-solving, which are skills that AI systems are increasingly able to replicate. By this reasoning, expectations of future improvement in AI may drive students away from programming careers.

However, once AI is as good at programming as humans, it will be able to quickly improve itself, potentially leading to a technological singularity in which humans are made obsolete. This is because AI systems will be able to design and program even more advanced AI systems, leading to exponential increases in their capabilities (Yudkowsky 2008). Therefore, it is possible that computer programming may be the last job done by

humans before AI surpasses human intelligence in all areas. Expectations of this kind of scenario may cause students to favor programming over other careers.

In general, expectations of future improvement in AI should cause students to attempt to estimate what careers will be safe for the longest period of time from obsolescence and spend resources to learn these careers. Ideally, students would develop the capacity to quickly develop the capacity to acquire new skills, but it's not clear that our educational system knows how to do this with anyone beyond teaching math and reading.

## POLITICAL JOB PROTECTION

It seems likely that different professions will attempt to use their political influence to prevent their skill sets from being made obsolete by AI (Hanson 2022). Doctors, for example, might try to ban diagnosis by AI, while lawyers might seek to prevent AIs from writing legal documents.

We can't be sure which if any neo-luddite job protection plans will succeed. But students should be more likely to attempt to enter a profession they think will have the political force to protect itself from AI competition.

Concentrated interests beat diffuse interests (Olson 2012) so an innovation that promises to slightly raise economic growth but harms, say, lawyers could be politically defeated by lawyers because they would care more about the innovation than anyone else. But, ignoring the possibility of unfriendly unaligned AI, AI promises to give significant net economic benefit to nearly everyone, even those whose jobs it threatens. Consequently, there will not be coalitions to stop it, unless the dangers of unaligned unfriendly AI become politically salient. The US, furthermore, will rightfully fear that if it slows the development of AI, it gives the lead to China, and this could be militarily, economically, and culturally devastating to US dominance. Finally, Big Tech has enormous political power with its campaign donations and control of social media, and so politicians are unlikely to go against the will of Big Tech on something Big Tech cares a lot about. Consequently, it seems reasonably likely that almost no profession, at least in the US, will succeed in protecting itself from AI competition.

## GREATER DESIRE OF GOVERNMENTS TO BORROW

Politicians as well are influenced by their expectations of the future. It's almost always in the short-term political interest of politicians to spend more money today and borrow funds to make up for the deficit. Of course, when this continually happens over a long period of time, a nation faces financial ruin. This expectation of future harm of borrowing too much today imposes some discipline on politicians. But what would happen if politicians figured that AI-powered technological change is going to render our society extraordinarily rich in twenty years? What would stop them from borrowing a huge amount now? These politicians would be in a similar position to the children of an extraordinarily wealthy and elderly parent who expect soon to inherit great riches. These children would be rational if they went into great debt expecting to pay off this debt with their inheritance. If, however, the parents end up giving their estate to charity, the children are financially ruined.

## ECONOMIES OF SCALE

Economies of scale refer to the cost advantages that a firm experiences when it increases production of a good or service. The concept is based on the idea that as a firm increases production, it can spread out its fixed costs over a larger number of goods, resulting in lower per-unit costs. This means that larger firms are typically more cost efficient compared to smaller firms.

In industries where large-scale operations are necessary to achieve cost efficiencies, firms that can take advantage of economies of scale may have a significant cost advantage over their smaller competitors. This can make it difficult for smaller firms to compete, as they may not be able to achieve the same level of cost efficiency. As a result, it is often the case that only a few firms can dominate an industry. These firms can use their scale to drive down costs and offer lower prices than their smaller competitors, making it difficult for other firms to gain a foothold in the market.

The AI industry is no exception to the impact of economies of scale. There are several sources of economies of scale in the AI industry, including data, specialized hardware, talent, research and development, and infrastructure. For example, firms with large amounts of data can achieve economies of scale in the development and deployment of AI systems, as more data leads to more accurate and effective AI systems.

Similarly, firms that can make large-scale investments in specialized hardware, such as graphics processing units, or in research and development, may be able to achieve economies of scale in the development and deployment of AI systems.

## TRADE BLOCKS

Imagine an alternate history in which Germany in 1938 stuck to the Munich agreement and so avoids war. But also imagine that Germany immediately engaged in a crash program to create atomic weapons, and both the US and Germany learn of each other's nuclear ambitions. It's likely the competition to be the first to create an atomic bomb would not only cause both nations to speed up nuclear progress as much as possible, but the countries would also seek to sabotage each other's programs in part through economic sanctions.

While modern communist China is certainly morally superior to Nazi Germany, it is still unfriendly with the US, and this makes it extremely unlikely that the US or China would want to see its rival develop technologies that gave this rival a decisive military advantage. If the US expects that AI will play a major role in military strength, and that economies of scale will dominate AI development, the US may seek to decapitate China's AI industry through restricting this industry's potential sales. The US might seek to form economic blocs that exclude China and China's allies. If American AI companies have the capacity to sell AI products to a far larger market than Chinese AI companies do, economies of scale would give these US companies an enormous advantage over their Chinese counterparts.

A striking feature of the modern world is the dominance of American tech companies in terms of size and influence. Many of the largest tech companies, such as Apple, Alphabet, and Microsoft, are based in the US. The expectation that AI will play a major role in the military balance of power and that economies of scale will be important in creating AI might cause the US to build on this advantage and force the rest of the world to either interact with US tech companies or Chinese ones, but not both.

Excluding Chinese AI companies from trade with a significant part of the world could have serious consequences for the global economy and the balance of power among nations. For example, if the US successfully limits China's access to key AI technologies, it could lead to a further

widening of the technological gap between the US and China. This could result in a more unbalanced distribution of economic power and influence, with negative consequences for countries that rely on China for trade or economic cooperation. On the other hand, if the US is unsuccessful in limiting China's access to AI technologies, it could lead to a more competitive and potentially volatile global AI market.

It is possible that the US tech industry may also seek to raise public fear about the potential risks of Chinese AI. For example, Facebook recently stoked controversy over TikTok, a Chinese-owned social media platform, politically pushing the claim that it poses a national security risk (Lorenz and Harwell 2022). While it is important to consider the potential risks of any new technology, it is worth noting that such concerns may also be motivated by the desire to exclude Chinese competitors from the market.

Russia's invasion of Ukraine has had significant consequences for the balance of power among nations, particularly in terms of military strength and economic cooperation. The invasion caused Europe to be more fearful of Russia, pushing European countries farther into the alliance with the US and NATO. This has strengthened the US's military position and its ability to exert influence over European countries. The invasion has made it far more likely that much of Europe would join a US economic bloc that excluded Chinese AI goods.

Taiwan becomes far more important if people expect AI and economies of scale in AI to play a huge role in future military outcomes. Taiwan is a major supplier of technology and plays a crucial role in computer chip production. However, its importance goes beyond just its economic contributions. Taiwan's strategic location in the Pacific also makes it a key player in the balance of power between the US and China. If China were to conquer Taiwan, it would give the Chinese Navy a significant advantage as they would no longer be confined to the "first island chain." Japan would then have to be concerned that China could impose costs on them, or potentially even blockades, if Japan aligned with a US anti-Chinese AI block. However, if the US successfully protects Taiwan, Japan is likely to feel more secure in the US military system and be more willing to join any trade area that the US creates.

One potential concern for Australia in a scenario where the US expects that AI will play a major role in military strength and that economies of scale will dominate AI development is that it may be forced to choose between joining the US economic bloc or the Chinese economic bloc. Australia has close economic ties with both the US and China and may

not want to risk damaging these relationships by aligning with one side over the other. However, if the US and China were to form rival economic blocs that excluded each other, Australia may feel pressure to choose one side to access advanced AI technologies and maintain its military security.

South Korea is a major source of technology and is very hostile toward Chinese-backed North Korea. In a scenario where economies of scale are thought to be important, China may seek to help North Korea conquer South Korea to bring it into the Chinese bloc. This could increase the dependence of South Korea on the US and make it more willing to follow rules set by the US regarding AI trade development.

The role of India as a swing player in this scenario cannot be underestimated. If the US expects that AI will play a major role in military strength and that economies of scale will dominate AI development, it may seek to bring India into its economic bloc to gain a larger advantage over China. India's position as a major player in the global economy, as well as its historical tensions with China, makes it a potentially valuable ally for the US in this scenario. However, India may also wish to maintain its independence and play off both the US and China for its own economic gain.

One strategy that India might employ is to trade with both the US and China, with the threat that if one trade bloc imposes sanctions on them, they will permanently shift their allegiance to the other bloc. This would allow India to negotiate better deals for itself and potentially extract more concessions from both the US and China. However, it is also possible that the US economic bloc will be larger and more attractive to India, and that India may be willing to join the US bloc and follow its rules in exchange for economic benefits.

A key concern for countries outside of the US that may be considering joining a US-led economic bloc centered on AI development is the potential for favoritism toward American companies. If the US were to prioritize the interests of its own firms over those of foreign companies, it could lead to a lack of fair competition and potentially disadvantageous terms for non-US firms.

One way that China could respond to US efforts to cut off its AI industry would be to prevent ethnic Chinese from working for US tech companies. Many software developers in the US are Chinese. China could demand that its citizens stop working for these companies and could even threaten punishment for those who have relatives in China and continue to work for US tech giants. It is worth noting that the US recently banned

US citizens from doing certain types of work at Chinese semiconductor fabrication facilities (Hill et al. 2022).

Ultimately, the future development of AI and its impact on global economics and military power is difficult to predict. However, expectations of the future of AI will shape the actions and decisions of nations, as they seek to position themselves for the changing geopolitical landscape.

## WHAT HAPPENS IF PEOPLE EXPECT MONEY TO BECOME WORTHLESS

Expectations of the future of AI progress will greatly harm the economy if many people expect that money and investments in the future will become worthless. Such expectations could arise if many come to believe in the view of AI researchers such as Eliezer Yudkowsky (2022) who think that AI will probably destroy us. On a more optimistic note, people could expect that money would become worthless because AIs would make us extraordinarily rich and so any additional money that we save today would be almost useless because of diminishing marginal returns to wealth or because AI will change society in a way that makes money no longer have value.

If people didn't expect money to have value in the future, they would have an incentive to draw down all their savings now and to not save for the future (Miller 2012, pp. 191–192). This would have a disastrous result on the ability of capital markets to raise funds. People would also drop out of the labor market if they could afford to survive until they thought that money would be worthless.

Paradoxically, thinking that money would become worthless at some future point would cause people to take actions that would push forward the point at which money would become worthless. Workers, for example, quitting their jobs as computer programmers or computer chip designers would make it harder for companies advancing AI to advance the field. This could happen because of very good or very bad expectations about what AI is likely to do.

This all could result in thought leaders having an incentive to manipulate expectations. If you believed that AI was likely to kill us, you would have an incentive to exaggerate the likelihood of it killing us to get people to drop out of labor markets to slow down economic progress. If you thought AI was going to do fantastic things and make us all rich, you

might downplay this hope for fear that workers would have an incentive to leave the job market if they came to believe in a coming AI utopia.

If people perceived that money would become worthless in the future, interest rates would have to become extraordinarily high. Imagine many believed there was a 60% chance that next year all savings would be worthless, for reasons good or bad. Companies, however, still wish to raise capital. To raise such capital companies would have to promise huge interest rates to draw in money to financial markets to compensate for the fact that people thought that money would probably be worthless. These high interest rates would of course make it challenging for companies and governments to borrow. Wages would have to greatly increase to compensate for people not wanting to save for the future. Lots of older working professionals have substantial retirement savings, and if they knew they would only be alive for a few more years, they might drop out of the workforce and draw down their savings.

## ANIMAL WELFARE

Expecting AI to soon have traits that most humans consider central to why we are morally distinct from animals creates a host of challenges (Bostrom and Yudkowsky 2018). As AI becomes more advanced, it is possible that societal expectations about the cognitive abilities and consciousness of non-human entities, such as animals, may change. This could potentially lead to a greater recognition of animal rights and a greater concern for their well-being.

As AI becomes more sophisticated, it may become increasingly difficult to justify the idea that humans are uniquely special or deserving of certain rights, while other intelligent beings (whether they be animals or artificial entities) are not. If non-human entities can exhibit intellectual and cognitive behaviors that are similar to or even surpass those of humans, it may become more difficult to justify their unequal treatment.

If expectations of AI reach a point where most everyone expects the AIs will be smarter than people, it will be difficult for us to justify mistreatment of animals on the assumption that we are much smarter than them because that will lead people to conclude that in the future AIs would be reasonable in not recognizing our “human rights” (Chapman 2017).

## CONCLUSION

Imagine a scenario where ChatGPT advances to a point where it can draft legal documents, leading to most law firms announcing a hiring freeze. Candid law professors advise their students that they will only have a lucrative career if laws get enacted forcing clients to use human lawyers. Additionally, radiologists now suggest their residents pursue alternative medical specialties as AI is predicted to outperform humans in radiology within the next ten years. People in most professions see the AI writing on the wall and know that even if they could squeeze out another decade in their career, it would be foolish for a young person to sacrifice to train for it. In this situation, tiger parents who in the past would have relentlessly pushed their children to excel at school now are mostly alright with their kids just having fun because they figure there's nothing the children are going to learn that will help them compete against future AIs. Furthermore, the kids might be having a lot more fun than before because AI-produced movies, books, and music are vastly better than anything we've previously had.

If ChatGPT and its successors convince many that AI will soon cause us to be living in changing times with a high variance of outcomes, people will behave in ways that we might have some success in forecasting. While forecasting how AI will develop (Grace et al. 2018; Makridakis 2017; Muller and Bostrom 2013; Barrat and Goertzel 2011) is more important than predicting how people will respond to such expected developments, the latter is likely substantially easier and worth extensively studying.

Much of our social and economic fabric comes from underlying expectations of how individuals, businesses, and governments will behave. These agents are behaving in ways to achieve good outcomes for themselves. Expected rapid advancement in AI progress will likely change the optimal self-interested behavior for these classes of agents. Unfortunately, many of the ways that we can predict people will change their behavior would make society as a whole worse off. Most troubling are that individuals and businesses could invest substantially less in the future and conflict between the US and China becomes more likely. Mistakes in estimating how much importance their great power rivals place on AI could exacerbate tensions as, for example, China might interpret US promoting its domestic computer chip industry as an act of willful military aggression, when in reality US politicians did it to slightly boost their economic growth or to reward high-tech campaign contributors.

Understanding the beliefs of various agents and identifying differences in those beliefs is crucial for effective decision-making. However, this becomes particularly challenging in the context of AI, as there is a lack of consensus regarding its potential outcomes. While some foresee significant progress with AI, others express concern about the loss of control to machines or even fatal consequences. This instability in beliefs poses a significant challenge for planning, as it increases the likelihood of radical norm-violating actions by individuals, firms, and nations. Thus, it is imperative to continue exploring the potential implications of AI while working toward a shared understanding of its future.

## REFERENCES

- Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint [arXiv:1606.06565](https://arxiv.org/abs/1606.06565).
- Barrat, J., & Goertzel, B. (2011). How long till AGI?—Views of AGI-11 conference participants. HPlusMagazine.com <http://hplusmagazine.com/2011/09/16/how-long-tillagi-views-ofagi-11-conference-participants/>.
- Bostrom, N., & Yudkowsky, E. (2018). The ethics of artificial intelligence. In *Artificial intelligence safety and security* (pp. 57–69). Chapman and Hall/CRC.
- Chapman, B. (2017). Animal Rights in the Age of Superintelligence What happens if humans become the animals? *Medium*, 13 June. [https://medium.com/@Ben\\_Chapman/animal-rights-in-the-age-of-superintelligence-47e335cf9a51](https://medium.com/@Ben_Chapman/animal-rights-in-the-age-of-superintelligence-47e335cf9a51).
- Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv preprint [arXiv:1702.08608](https://arxiv.org/abs/1702.08608).
- Grace, K., Salvatier, J., Dafoe, A., Zhang, B., & Evans, O. (2018). When will AI exceed human performance? Evidence from AI experts. *Journal of Artificial Intelligence Research*, 62, 729–754.
- Hanson, R. (2022). Twitter 21 Dec 2022 8:42 a.m. <https://twitter.com/robinhanson/status/1605559382581735425>.
- Hill, S., Ettinger, S., Zaucha, J., Christensen, G., Orenstein, J., & Johnson, C. (2022). U.S. Government imposes significant new export controls on semiconductor, semiconductor manufacturing equipment, and supercomputer-related transactions involving China and Chinese entities, 21 October. KLGates.com. <https://www.klgates.com/US-Government-Imposes-Significant-New-Export-Controls-on-Semiconductor-Semiconductor-Manufacturing-Equipment-and-Supercomputer-Related-Transactions-Involving-China-and-Chinese-Entities-10-21-2022>.

- Hubinger, E., van Merwijk, C., Mikulik, V., Skalse, J., & Garrabrant, S. (2019). Risks from learned optimization in advanced machine learning systems. arXiv preprint [arXiv:1906.01820](https://arxiv.org/abs/1906.01820).
- Lipton, Z. C. (2016). The Mythos of Model Interpretability. arXiv preprint [arXiv: 1606.03490](https://arxiv.org/abs/1606.03490).
- Lorenz, T., & Harwell, D. (2022). Facebook paid GOP firm to malign TikTok. *Washington Post*, 30 March. <https://www.washingtonpost.com/technology/2022/03/30/facebook-tiktok-targeted-victory/>
- Makridakis, S. (2017). The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. *Futures*, 90, 46–60.
- Miller, J. D. (2012). *Singularity Rising: Surviving and thriving in a smarter, richer, and more dangerous world*. BenBella Books, Inc.
- Muller, V. C., & Bostrom, N. (2013). Future progress in artificial intelligence: A survey of expert opinion. In *Fundamental issues of artificial intelligence*. Synthese Library. Berlin: Springer.
- Olson, M. (2012). The logic of collective action [1965]. Contemporary Sociological Theory, 124.
- Sandberg, A. (2022). Twitter 27 Dec 2022 5:29 p.m. [https://twitter.com/and\\_erssandberg/status/1607866411732602881](https://twitter.com/and_erssandberg/status/1607866411732602881).
- Yampolskiy, R. V. (2020). On Controllability of AI. arXiv preprint [arXiv:2008.04071](https://arxiv.org/abs/2008.04071).
- Yampolskiy, R. V. (2022). Posted to Yampolskiy's Facebook page on December 9. Accessed January 13, 2023.
- Yudkowsky, E. (2008). Artificial intelligence as a positive and negative factor in global risk. *Global Catastrophic Risks*, 1(303), 184.
- Yudkowsky, E. (2022). MIRI announces new "Death With Dignity" strategy. LessWrong.Com, 1 April. <https://www.lesswrong.com/posts/j9Q8bRmwCgXRYAgcJ/miri-announces-new-death-with-dignity-strategy>. Accessed January 13, 2023.

PART II

---

The Status of Machine Learning Methods  
for Time Series and New Product  
Forecasting



## CHAPTER 3

---

# Time Series Forecasting with Statistical, Machine Learning, and Deep Learning Methods: Past, Present, and Future

*Evangelos Spiliotis*

## INTRODUCTION

The uncertainty about the future and the complexity of business environments have brought forecasting at the forefront of decision making. From short-term operational decisions to long-term strategic ones, accurate forecasts are required to facilitate planning, optimize processes, identify risks, and exploit opportunities. Consequently, forecasters have devoted significant efforts in the development of new forecasting methods and the improvement of existing ones (Petropoulos et al., 2022a).

This chapter provides an overview of the key classes of methods used in the field of time series forecasting. By reviewing the properties of each

---

E. Spiliotis (✉)

Forecasting and Strategy Unit, School of Electrical and Computer Engineering,  
National Technical University of Athens, Athens, Greece

e-mail: [spiliotis@fsu.gr](mailto:spiliotis@fsu.gr)

class of methods, the chapter summarizes their main strengths and weaknesses and explains the conditions under which they may perform better or at least be the most reasonable alternative to choose from. In addition, it discusses recent methodological advances that can be used to improve forecasting accuracy and proposes some directions for future innovations.

The insights provided by the chapter offer several practical benefits, being relevant both for forecasting researchers and for practitioners. For instance, apart from explaining how statistical, machine learning (ML), and deep learning (DL) methods work in practice and how they may successfully be applied for time series forecasting, the chapter introduces, based on its observations, some subjective criteria that can be used for selecting a method of preference according to the characteristics of the data set at hand (e.g. in terms of data availability, cross-sectional relationships, explanatory variables, and nonlinear relationships), the uncertainty present, and the core objectives of the forecaster (e.g. in terms of accuracy, computational cost, development effort, and forecast explainability). In addition, it summarizes some useful and relatively unexplored techniques that can be employed to further improve the performance of statistical, ML, and DL methods or even combine their strengths. Thus, good practices are effectively highlighted and bad ones are discouraged.

## STATISTICAL METHODS

In their seminal paper, De Gooijer and Hyndman (2006) review time series forecasting practices, covering all the major types of forecasting methods with a focus, due to the year the study was published, on what are nowadays considered “statistical”<sup>1</sup> forecasting methods. This includes exponential smoothing, ARIMA, seasonal, and state space models, among others. What these methods have in common is that they prescribe the underlying data-generating process, making assumptions about the structural components of the series, and the way historical observations are correlated (Barker, 2020). For instance, a method may assume a damped trend, an additive seasonality, normally distributed forecast errors, or a linear relationship between succeeding observations to produce forecasts.

Although this prescriptive forecasting approach may sound restrictive, it has been proven particularly efficient in various practical applications

<sup>1</sup> Other terms commonly used to describe statistical methods are “traditional”, “conventional”, “structured”, “model driven” or simply “time series” methods.

for many decades (Makridakis et al., 2018). The main advantage of statistical methods is that they typically involve a limited number of parameters. Hence, they can adequately fit time series data, even in cases where the available historical observations are limited. Moreover, in such modeling settings, parameter uncertainty becomes less relevant, issues related to overfitting are mitigated, while computational cost is usually small. In addition, forecasting can be automated and executed on a regular, systematic basis, while predictions are intuitive and easy to communicate to decision makers. These properties facilitate forecasting at scale and allow the easy application of forecasting methods in businesses and organizations that may not necessarily have sufficient data, computational resources, or human capital to invest in the process (Petropoulos et al., 2021).

Nevertheless, the simplicity of statistical methods comes at a price. As explained, given a certain statistical method, the model to be used for producing forecasts will be prescribed, meaning that selecting the “wrong” model may result in inaccurate forecasts, especially in applications where patterns are nonlinear or change irregularly. Unfortunately, selecting the most appropriate method is not a trivial matter. First, the “true” model that sufficiently mimics the underlying data-generating process of the series always remains unknown to the forecaster, rendering its identification impossible. Second, there are numerous models available to choose from to approximate what may be believed to be the true model. Yet, considering all possible models when performing a selection is impractical. Third, even if a certain pool of candidate models is considered, identifying the most accurate one is challenging due to the uncertainty involved. Simply put, the nuances of the past may not persist into the future, while the nuances of the future may not have been revealed in the past (Tashman, 2000). Petropoulos et al. (2018) found that tackling model uncertainty brings the most benefits in statistical forecasting, even higher than mitigating data or parameter uncertainty, a conclusion that clearly demonstrates the challenges involved in forecasting model selection.

To deal with model uncertainty, forecasters have been combining (or ensembling) different models (Bates & Granger, 1969), a practice that typically results in more accurate forecasts than using a single model. Forecast combination builds exactly upon the notion that “all models are wrong, but some are useful”, a famous aphorism attributed to George Box. In order for the ensemble to add more value, it is usually

recommended that the base forecasts are as many and as diverse as possible (Kang et al., 2022). Interestingly, although many sophisticated approaches have been proposed to optimally combine forecasts (Montero-Manso et al., 2020), the practice shows that, due to the uncertainty present when estimating the combination weights, simple operators (e.g. mean or median) are likely to provide similarly accurate, if not better results (Petrooulos & Svetunkov, 2020). This phenomenon is referred to as a “forecast combination puzzle” and is widely investigated in the forecasting literature with the objective to further improve performance (Claeskens et al., 2016). The results of various studies, including popular forecasting competitions (Makridakis et al., 2020), showcase the value added by ensembling and motivate its use.

Aside from ensembling, many innovations have been proposed to enhance the performance of statistical methods through the years. A major dimension in this direction involves utilizing techniques that manipulate the data before applying any statistical model, allowing the extraction of more information or the mitigation of undesired issues related with data, model, and parameter uncertainty (Petrooulos & Spiliotis, 2021). Thus, data processing can be regarded as a “self-improving mechanism” (Spithourakis et al., 2011) that boosts the performance of the underlying statistical models. For instance, power transformations (e.g. logarithmic or Box-Cox) on the time series data (Proietti & Lütkepohl, 2013), transformations on the forecast errors of the models (Beaumont, 2014), and smoothing techniques (Spiliotis et al., 2019) can be applied to reduce data variance and allow linear models to properly fit nonlinear series. The local curvatures of the series can also be adjusted to enable the models to focus either on the short- or long-term behavior of the data depending on the objectives of the forecasting task at hand (Assimakopoulos & Nikolopoulos, 2000). Improved seasonal adjustments (Miller & Williams, 2003) and forecasts based on sub-seasonal series (Li et al., 2021) can simplify pattern analysis and offer more robust estimations. Bagging (bootstrapping and aggregation) and other data augmentation (or oversampling) techniques can be employed to resample the series toward the creation of multiple new ones with the same underlying patterns but different randomness with promising results (Bergmeir et al., 2016). Finally, multiple temporal aggregation can be used to combine forecasts produced using data of transformed series, each expressed at a higher frequency than the original (Athanasopoulos et al., 2017). Given that high frequency series highlight components like

seasonality and noise, while low frequency series emphasize components like level and trend, identifying and modeling complex patterns becomes easier under the multiple temporal aggregation framework.

Another dimension toward the improvement of statistical methods is the extension of their original features. Notable innovations in this direction include the development of robust, automated selection algorithms that can choose the most appropriate model from a family of models using information criteria. These frameworks have further popularized the use of exponential smoothing (Hyndman et al., 2002) and ARIMA (Hyndman et al., 2020) models, rendering them generic forecasting solutions. State space modeling approaches, providing a unifying framework in which any linear time series model can be expressed, have also contributed to that end (Kalman, 1960). Significant work has also been conducted on the use of nonlinear models (De Gooijer & Kumar, 1992) and the exploitation of Monte Carlo simulations and bootstrapping, aiming to better adapt statistical methods to more real-life applications. Although nonlinear statistical models have shown some potential, making less assumptions about the distribution of the data, the accurate estimation of their parameters, efficient implementation, and robust computation of forecasts remain open issues. Moreover, generalizations and modifications of successful statistical models, like Theta (Spiliotis et al., 2020) and exponential smoothing (Svetunkov et al., 2022), have been proposed to improve their performance for various types of series, including nonstationary processes and nonlinear trends. Finally, forecasters have tried to capitalize on large data collections, developing methods that produce forecasts based on the similarity of the target series with other reference series (Kang et al., 2021) or the general tendencies of the models in terms of realized forecasting performance (Petrooulos et al., 2022b).

A special note shall finally be made on the performance of statistical methods when it comes to probabilistic forecasting, i.e. the estimation of quantile forecasts, prediction intervals, or complete probability distributions. Pure statistical models produce probabilistic forecasts analytically by processing the residual errors and making certain assumptions about their distribution. Although this approach is convenient, requiring little or no additional computations after fitting the model, it is limited in the sense that the residuals are typically assumed to be normally distributed, also referring to one-step-ahead forecasts (Kolassa, 2016). As a result, when the model is to be used for multi-step-ahead forecasting or for predicting

series whose data are abnormally distributed, the forecasts may considerably underestimate uncertainty (Makridakis et al., 1987). Therefore, more recent approaches are based on empirical computations, bootstrapping, and simulations (Spiliotis et al., 2021) that may account for critical factors like the forecasting horizon, the distribution of the data, and the uncertainty about the future (out-of-sample forecast errors instead of in-sample ones). These techniques are also used for statistical models where there are no known formulas for producing probabilistic forecasts and methods that are not based on certain model forms.

## MACHINE LEARNING METHODS

As explained, despite their notable strengths, statistical methods come with some limitations, the most critical being their tendency to make strong assumptions about the data-generating process, the employment of mostly linear model forms, and their finite learning capacity in terms of estimated parameters. To tackle these issues, forecasters have considered the application of nonlinear regression methods that have successfully been used in the field of data science to solve complex pattern recognition and image processing problems, among others (Makridakis et al., 2018). These methods are typically referred to as “machine learning”<sup>2</sup> and have revolutionized forecasting theory and practice, providing excellent results in various settings (Zhang et al., 1998). The most representative and broadly used ML methods for time series forecasting are the neural networks (NNs, Hewamalage et al., 2021) and the regression trees (RTs, Januschowski et al., 2022), but other algorithms like K-nearest neighbors regression (KNNR) and support vector regression (SVR) have also been considered by forecasting researchers.

A fundamental advantage of ML methods is that they make few or no assumptions<sup>3</sup> about the target series, allowing for data relationships to be identified and estimated automatically, thus being more generic. As a result, the challenging task of selecting the most appropriate forecasting model becomes less relevant and the focus of the forecaster turns

<sup>2</sup> Other terms commonly used to describe ML methods are “computational intelligence”, “unstructured”, or “data-driven” methods.

<sup>3</sup> Although it is true that some ML models make certain assumptions about data distributions, they rarely prescribe the structural components of the series.

to optimally designing the training process of the selected model and estimating the values of the hyperparameters controlling said process. This “self-learning” property is mainly supported by the nonlinear mechanisms and empirical optimization algorithms the ML methods utilize to produce forecasts, including nonlinear activation functions (the sigmoid, hyperbolic tangent, and rectified linear unit functions are commonly used in NNs to compute the output of a node given an input), gradient-based optimization methods (e.g. implemented in NNs and boosted trees), piecewise regression techniques (RTs forecast by splitting the samples of the train set into multiple subsets based on certain rules applied to the regressor variables and fitting a separate line segment in each subset), similarity-based estimations (KNNR forecasts by identifying the K most similar instances to the target one and computing their average value), and kernel-based transformations (polynomial, sigmoid, and radial basis function kernels are used in SVR to map nonlinear combinations of the predictor variables into a higher dimensional space), among others.

Another valuable property of ML methods is their increased learning capacity, i.e. their ability to produce forecasts by blending numerous predictor variables, while utilizing a large number of parameters. Therefore, ML methods can effectively handle large dimension data sets and analyze complex data relationships to project them into the future. Despite that in theory some statistical models have similar properties, in practice, it is very challenging or computationally expensive to implement them. For instance, although it is true that ARIMA models can include several autoregressive and moving averages terms, the analytical estimation of their coefficients may become infeasible at some point, leading to unstable or inaccurate forecasts. Similarly, although linear regression methods can include numerous predictor variables and even automatically identify the most relevant ones (e.g. lasso and ridge regression can be used to reduce or completely remove the effect of relatively irrelevant predictor variables), the maximum number of their coefficients will still be subject to the number of the predictor variables considered, thus having lower learning abilities compared to ML methods.

The improved learning capacity of ML methods can also be exploited to develop “global”, generalized forecasting models that provide accurate forecasts for multiple time series simultaneously (Januschowski et al., 2020). The vast majority of statistical methods have been designed to

produce forecasts for a single series at a time.<sup>4</sup> Consequently, a separate, “local” model has to be fitted to each series and used for extrapolation. In contrast, by design, and given that they build upon regression techniques that process tabular data, ML methods can be modified to learn from multiple series how to best forecast the individual ones. Effectively, global models assume that all series follow the same data-generating process and that each series contributes independent samples of this process during training. This concept, commonly referred to as “cross-learning”, has recently reported particularly promising results, attracting the attention of both researchers and practitioners (Semenoglou et al., 2021). In fact, for many years, there was a misconception that ML methods were inappropriate for time series forecasting, a belief that was mostly driven by the less mature algorithms available at the time, but also by the employment of local training strategies (Makridakis et al., 2018). The popularization<sup>5</sup> of global training strategies (Smyl, 2020) signified a major change in ML time series forecasting and, since then, cross-learning is considered best practice and the natural way of applying ML forecasting methods. The potential benefits of using global ML models over local ones are several and essential (Montero-Manso & Hyndman, 2021). First, when forecasting at scale, it is typically computationally more efficient to train a single global model than fit multiple local ones, a practice that can also contribute to lower operational costs (Makridakis et al., 2018). Second, the effort needed to develop, monitor, and maintain a global model is expected to be lower than that of local approaches, each requiring separate optimization and review. Third, local models cannot

<sup>4</sup> Vector ARIMA (VARIMA) and its variants, providing a multivariate generalization of the univariate ARIMA model, are probably the most notable counterexamples. Yet, their modeling approach still differs from global models in the sense that they are not tasked to forecast each series separately, but accommodate assumptions on cross-series and contemporaneous relationships. Also, when the number of series becomes large, VARIMA can be challenging to estimate and include many insignificant parameters, thus becoming impractical (Riiise & Tjozstheim, 1984). Similarly, although one can argue that hierarchical forecasting methods (Hyndman et al., 2011) allow the exchange of information between multiple series, they fundamentally differ from global models as said exchanges are possible just for hierarchically organized data and not for series that may originate from different data sources or represent a variety of measures.

<sup>5</sup> Although some primitive global models have been proposed before 2018 (e.g. a NN model that was trained on pools of series was ranked 3rd in the NN3 competition; Crone et al., 2011), the M4 competition (Makridakis et al., 2020) was probably the first major study to demonstrate their superiority.

capture possible correlations between different series, which may be a useful modeling feature for better learning data relationships and identifying factors that affect the target series. In the Big Data era, where large, high frequency data sets consisting of multiple correlated series can be constructed, information exchange can undoubtedly offer significant advantages to forecasting models. Fourth, global models are more likely to provide accurate forecasts for series that consist of few observations as, in contrast to local models, their predictions will be supported from the knowledge acquired by analyzing other, potentially longer series of similar characteristics. Fifth, given an adequately generalized global model, its knowledge can be used for zero-shot forecasting (or transfer-learning), i.e. for predicting series that were not originally available in the train set (Wellens et al., 2022). The latter two properties can become handy in settings where the target series naturally consist of few observations or new series are regularly added to the data set, as is the case in retail and new product forecasting (Spiliotis et al., 2022). As an all-welcome side-effect of the above properties, computational effort can be further reduced, while pre-trained ML models can be publicly made available to democratize the use of sophisticated forecasting methods (Chatigny et al., 2021).

The design of ML methods also provides a convenient framework for including exogenous (or explanatory) variables as predictors in addition to the historical observations (autoregressive terms or lags) of the series. In univariate time series forecasting settings, the past observations of the target series consist the core source of information provided to the models as input<sup>6</sup> for making predictions. As a result, most forecasting models are ultimately based on some sort of autoregression, using either the raw historical observations or some products of those (e.g. statistics computed on historical observations or residual errors) as input. ARIMA models are indicative examples of this training approach, but most ML forecasting methods will also employ similar strategies to identify patterns and/or relationships in the data (Smyl, 2020). In fact, a feedforward NN that uses an identity (linear) activation function and a look-back window of length  $p$  would be equivalent to an AR( $p$ ) model (Barker, 2020), with their only difference being on the way each method computes the autoregressive coefficients (maximum likelihood estimation

<sup>6</sup> The term “look-back window” is typically used to define the number of lags considered by the model for making forecasts.

versus backpropagation). Yet, exogenous variables can complement historical data, accounting for factors that affect the target series and distort their patterns (Ma et al., 2016). For instance, variables that represent holidays, special days, promotions, prices, and weather conditions can be used to improve the accuracy of sales forecasting models (Makridakis et al., 2022a). With few exceptions, such as ARIMAX models, statistical methods cannot include exogenous variables as the analytical estimation of the respective coefficients can become imprecise or too challenging to conduct, also resulting in unstable forecasts (e.g. this is the case for exponential smoothing models). This is particularly true when the number of the exogenous variables becomes significant. In contrast, this process can be easily applied through the regression frameworks the ML methods build upon, enabling the embedment of various features in an efficient manner. Moreover, due to their learning process, most ML methods offer mechanisms that automatically select the most vital features (or diminish the effect of irrelevant ones), an essential property<sup>7</sup> when the total number of the predictor variables grows in size.

Of course, ML methods also come with some notable drawbacks and should not be oversold as miracle forecasting approaches. First, since ML methods tend to make no assumptions about the data, sufficient data availability is a prerequisite for them to be effectively employed, especially when the number of available predictors is large. The “curse of dimensionality”, commonly used as a term to explain that the amount of data needed to obtain reliable results grows exponentially as the dimensions of the provided input increase, is a key limitation of ML methods and particularly of the NNs that may involve thousands of trainable parameters. Although modern data sets can in many cases be large enough<sup>8</sup> to enable the generalization of the models, over-parametrization and

<sup>7</sup> As an example, in each iteration, a RT will split the samples of the train set using a rule that applies on a certain regressor variable so that the forecast error is minimized. In this manner, given a set of predictors that includes both relevant and irrelevant regressor variables, only the former will be used. In the same forecasting task, given a feedforward network, the weights of the nodes that correspond to the irrelevant predictors will gradually be set to zero to minimize their impact on the forecast. Although in practice the above claims may not be absolutely true, it is evident that, structurally speaking, the ML methods do allow an automated feature selection process.

<sup>8</sup> Larger data sets can be created either by including longer series, each contributing multiple observations to the train set, or by including several short series, together contributing numerous observations to the train set.

overfitting can still negatively affect forecasting performance, as they are also issues that are difficult to identify when initially designing and training the models. As a result, it is not surprising that in many forecasting applications statistical methods are still found to outperform ML approaches (Makridakis et al., 2018). In this respect, data augmentation techniques (Semenoglou et al., 2023a) and advanced cross-validation strategies (Lainder & Wolfinger, 2022) have been proposed to mitigate the impact of insufficient data, overfitting, data leakage, and lack of generalization in forecasting tasks that involve nonstationary and collinear series.

Another practical issue with ML methods is their greater development cost. In contrast to statistical methods, that can nowadays be automatically optimized in a relatively straightforward and fast manner using any modern software of preference (Hyndman et al., 2020), constructing a ML method remains more of an art. This is because most ML methods, and especially NNs, consist of several elements and utilize a variety of hyperparameters that have to be carefully defined or optimized in order for the model to produce accurate results (Bojer, 2022). In addition, data processing and manipulation (e.g. feature engineering and data transformations) is more relevant to ML methods compared to statistical ones, highly affecting their forecasting performance. Therefore, the time and effort required for building and validating ML methods are more significant compared to standard statistical methods, with the realized accuracy also being subject to the experience of the data scientist. This may partially explain why relatively simpler ML methods, like tree-based models, are more widely used in practice than NNs (Bojer & Meldgaard, 2021), despite the latter being similarly capable of handling time series forecasting tasks. Tools that automate the selection, design, training, and validation of ML methods (see, for example, AutoKeras and AutoML) may mitigate the aforementioned limitations in the future (Makridakis et al., 2022b). However, there is still much work to be done in this direction to successfully achieve their objective, also keeping in mind that such tools will most likely lead to less intuitive and explainable models.

The relatively more complex structure of ML methods can also have a negative impact on the time required for computing the forecasts and, consequently, the monetary cost of forecasting. Even when global models are used to simultaneously forecast multiple time series, i.e. a single model is trained instead of a separate one per series, their training

time may still surpass that of fitting multiple, local statistical models (Petropoulos et al., 2021). This concern may not necessarily be relevant when interested in forecasting a limited number of time series once per quarter or year, but can become significant when forecasting frequently or at scale. Indeed, when it comes to large organizations like retailers and tech companies, computational time may become a critical factor for choosing which method to employ, especially when the expected accuracy gains of the more computationally intensive methods do not fully justify their use or the time available for generating the forecasts is limited by operational constraints (Gilliland, 2020). A counterargument for preferring a computationally more expensive ML method is that the major cost of training is paid only once and that once this process is completed, inference can be performed much faster. Although this is true, the final cost will effectively depend on how often the model will have to be retrained so that it takes into account the most recent data and maintain the expected level of forecasting accuracy. That said, global ML methods can become efficient in applications that do not require frequent model updates.

Similar to many statistical methods that cannot analytically derive probabilistic forecasts, ML models require special modifications to estimate the uncertainty around their point forecasts (Spiliotis et al., 2021). A popular approach to address this challenge is to directly task a new ML model to compute forecasts for a certain quantile of interest. To do so, the training setup of the model must be adjusted so that its forecasting performance is measured by an appropriate loss function, such as the pinball loss. The main drawback of this approach is that a separate, specialized model must be trained for each quantile, thus increasing the overall computational cost in applications where forecasts for multiple quantiles must be computed. Another approach is to record the forecast errors produced by the ML model used for producing point forecasts (e.g. using cross-validation) and empirically estimate the underlying uncertainty. Although this approach does not require training a separate model per quantile, it builds on simulated forecast errors that may be computationally expensive to acquire. More importantly, in contrast to the first approach, this technique cannot always<sup>9</sup> dynamically adjust its forecasts according to the variables provided to the model as input and which may simulate

<sup>9</sup> Some researchers have proposed focusing on similar past forecast errors (Shrestha & Solomatine, 2006) or point forecasts (Pinson & Kariniotakis, 2010) to the period being

factors that affect the degree of uncertainty (conditional forecasts). A third approach is to assume a particular data distribution for the series being forecast and task a ML model to estimate the parameters of said theoretical distribution (Salinas et al., 2020). The great advantage of this approach is that a single model can be used to provide both point and probabilistic forecasts of any quantile of interest, with the forecasts also accounting for the factors that possibly affect forecast uncertainty. On the other hand, identifying the distribution that best describes the target series is challenging, while the selected distribution may not be appropriate for the complete set of series, leading to sub-optimal results.

## DEEP LEARNING METHODS

Although ML methods have great learning abilities and they tend to provide more accurate results when more data is available for training, at some point, their performance will inevitably reach a plateau. This happens because depending on the number of trainable parameters the models have, they can only effectively handle up to a certain amount of data. To make ML models more scalable and allow their performance to improve as more data is provided as input, researchers have introduced the concept of DL.

Despite the fact there is no strict way to define DL, it can be regarded as a special class of ML based on NNs in which multiple processing layers are used to extract higher-level information from the data. Note, however, that constructing NNs which consist of multiple similar layers (e.g. training a feedforward NN of five dense layers) cannot really enable such processing. In practice, the architecture of the NN has to be carefully designed so that each layer (or stack of layers) progressively processes the data, extracting new features compared to the previous layers and, as a result, significantly contributing to the overall learning process.

In time series forecasting settings, the most indicative example of DL methods is probably N-BEATS (Oreshkin et al., 2019). In brief, the model builds on multi-layer feedforward NNs, called blocks, that are tasked to compute both forecasts and the residual errors of said forecasts. These blocks are organized into stacks using a doubly residual stacking

forecast with the objective to account for the effect that critical explanatory variables and seasonality have on forecast uncertainty, while reducing computational cost.

principle. The forecasts produced by the stacks are aggregated to provide the final forecasts. Observe that although the dense layers used by N-BEATS are the most common NN topology, the fact that each block learns how to best forecast given both previous predictions and residuals, enables the development of a truly DL and self-improving forecasting approach. DeepAR (Salinas et al., 2020) is also considered a pioneer in the field as it popularized through the GluonTS toolkit (Alexandrov et al., 2019) the use of deep autoregressive recurrent NNs (RNNs) for probabilistic forecasting.

Other interesting examples of DL include models that blend convolutional NNs (CNNs) with mostly RNNs. CNNs have been initially proposed for image processing as they involve special filters that analyze the input variables and extract higher-level features from the data, thus facilitating the identification of hidden patterns and correlations. However, recent studies have indicated that CNNs can complement or even outperform RNNs on sequence modeling tasks, making them relevant for time series forecasting (Bai et al., 2018). The work of Lai et al. (2017), mixing CNNs and long short-term memory (LSTM) NNs, is very indicative. Effectively, the proposed model can both extract short-term dependencies among variables and discover long-term patterns, a property that can help tackle the limitation of RNNs to account for multiple seasonal cycles (Hewamalage et al., 2021). In a similar direction, Shih et al. (2019) introduced a novel attention mechanism that enables the model to extract temporal patterns and focus on different time steps for different time series. Borovykh et al. (2017) adapted the architecture of WaveNet to introduce stacks of dilated convolutions that allow the model to access a broad range of history when forecasting and exploit the correlation structure between multiple series. More recently, Semenoglou et al. (2023b) proposed a DL method for univariate time series forecasting that mixes convolutional and dense layers in a single NN, using as input the visual representations of the series (images) instead of numerical vectors.

There is currently more than enough evidence to justify the use of DL methods and motivate their further development, especially in applications like energy, financial, and sales forecasting where large data sets can be naturally acquired. For example, in the M5 accuracy competition (Makridakis et al., 2022a), the 2nd and the 3rd winning submissions were built on variants of DeepAR and N-BEATS models, suggesting that DL models are on par, if not better than state-of-the-art ML approaches. Similarly, in their comparative study that covered a large set of diverse

series and four different DL models, Makridakis et al. (2022b) concluded that DL methods can perform better than most standard methods, particularly for longer forecasting horizons. However, this conclusion was mostly true when multiple DL models were combined. When this was not the case, most individual DL models were outperformed by simpler statistical and ML methods. This finding is in agreement with the results of Oreshkin et al. (2019) where N-BEATS was found to improve accuracy over the benchmarks only when ensembles of many<sup>10</sup> DL models were used. It seems that parameter uncertainty, over-parametrization, and overfitting still negatively affect the performance of DL models and that more work is required until forecasters will not have to rely on heavy ensembling to ensure high accuracy and robustness.

In terms of drawbacks, DL methods have similar weaknesses to those reported for ML methods, with the main difference being that training a DL model would typically demand significantly more data and computational resources (e.g. in terms of memory and training time). Moreover, the successful implementation of DL models will require using additional techniques that mitigate issues related to vanishing gradients and degradation of training accuracy (He et al., 2016). For instance, this includes the introduction of “shortcut connections” between the layers that enable their inputs to be directly transferred to their outputs when data processing is unnecessary (Smyl, 2020) or randomly omitting units (e.g. DropConnect and DropOut) during training (Wan et al., 2013). Other mechanisms include the employment of rectified activation functions<sup>11</sup> that only saturate in one direction (Agarap, 2018), adaptive

<sup>10</sup> This includes NNs that consider different weight initializations, sizes of look-back windows, and loss functions.

<sup>11</sup> The rectified linear unit (ReLU) and leaky ReLU are commonly used as activation functions in DL models.

gradient descent algorithms<sup>12</sup> (Kingma & Ba, 2014), batch normalization<sup>13</sup> (Ioffe & Szegedy, 2015), improved weight initializations (Glorot & Bengio, 2010), and greedy layer-wise training<sup>14</sup> (Bengio et al., 2006).

## DISCUSSION AND FUTURE STEPS

By reviewing the strengths and weaknesses of each class of time series forecasting methods, it becomes clear that no forecasting approach can consistently outperform the rest and that depending on the application, different methods may be more suitable (Petropoulos et al., 2014). In this regard, and when conducting objective, empirical comparisons (e.g. cross-validation) is not a favorable option, method selection could be based on the following subjective criteria.

1. **Data availability:** Most ML methods will typically require a minimum of a few thousand samples to be sufficiently trained, while more sophisticated models (e.g. deep NNs) may require significantly larger data sets (e.g. millions of samples). Therefore, if the available data set does not meet these requirements, proceeding with some statistical, local method would be more reasonable. In any case, the complexity of the selected method in terms of trainable parameters or rules should be proportional to the size of the train set at hand.
2. **Cross-sectional relationships:** If the data set consists of multiple series that are related in a cross-sectional fashion (e.g. products sold at different stores or energy demand reported for different areas), selecting a global ML or DL model would be the most promising approach, allowing for cross-series relationships and general tendencies to be effectively learned. Note that accurate global models can be trained even when the series of the data set are not directly related but share some key principles (e.g. have the same data frequency or

<sup>12</sup> The root mean squared propagation (RMSProp) and the Adam optimizers can be used to dynamically adapt the step size (learning rate) for each input variable based on the most recently observed gradients of said variable over the search.

<sup>13</sup> According to this approach, the inputs of the layers are normalized by re-centering and re-scaling so that the distribution of each layer's inputs does not vary during training.

<sup>14</sup> According to this approach, the layers of the NN are successively added to the model and refitted, allowing the newly added layers to learn the inputs from the existing ones.

originate from the same domain). However, in the latter case, the expected benefits of cross-learning should be smaller, while additional data processing may be required to better homogenize the individual samples.

3. **Explanatory variables:** In general, ML and DL methods provide more efficient frameworks than statistical models for incorporating explanatory variables in time series forecasts. However, depending on the type and number of features available, different ML or DL methods may prove to be more appropriate. For instance, decision-tree-based models are usually better at handling numerous features, especially when part of the explanatory variables is irrelevant to forecasting or their type is of a categorical nature. On the contrary, NNs are typically more effective at processing continuous explanatory variables that display nonlinear relationships with the target variable.
4. **Forecast uncertainty:** When the uncertainty around the forecasts is significant, investing in sophisticated ML or DL models may not always result in superior predictions. For instance, this may be the case when forecasting noisy data (e.g. financial series, disaggregated sales, or 15-min inbound calls) or interested in forecasting extreme quantiles (e.g. 0.995 or 0.999). In such settings, simple statistical models and empirical estimations may prove more useful and robust.
5. **Nonlinear data relationships:** A major advantage of the ML and DL forecasting methods is that they make no or few assumptions about the data and exploit nonlinear mechanisms to learn how series and observations are correlated. As such, ML and DL methods should be preferred over statistical models when working with abnormal series or data that are nonlinearly correlated. When this is not the case, simpler statistical methods could be used instead to avoid unnecessary complexity.
6. **Computational cost:** Before putting any forecasting method to use, forecasters should clearly define how often the model will be utilized to produce new forecasts (forecast cycle) and retrained to ensure adequate performance (update frequency). If the update frequency is relatively long or the computational cost for re-training the model is reasonable, ML and DL methods can offer affordable forecasting solutions. Otherwise, the potential accuracy improvements of ML and DL methods may just not be enough to justify their use on a regular basis, rendering the utilization of statistical models more

attractive. Nevertheless, since the computational cost of local statistical models is expected to increase linearly with the number of series being forecast, global models are competitive when forecasting at scale.

7. **Development effort:** As discussed, given their simplicity, statistical methods are usually straightforward to implement, requiring minor modifications. On the contrary, ML methods have to be carefully designed and tuned in terms of architecture, hyperparameters, and features used. This additional effort, which becomes more relevant for DL models, may be a critical factor for deciding which forecasting approach should be selected and if the expected accuracy improvements surpass the cost of model development.
8. **Forecast explainability:** Explainability refers to the ability of the model to justify the generated forecasts given its parameters, rules, or general structure. Although in many forecasting applications this property may be irrelevant, it is evident that when the forecasts are to be used to support critical decisions, explainability can become of significant importance, justifying possible risks taken. The forecasts produced by most statistical models are easy to explain and interpret since they prescribe the data-generating process, assume linear relationships, and involve a finite number of parameters. This is not always true for ML and DL methods, although interesting efforts have been made recently to enhance their explainability (Rajapaksha et al., 2022).

In addition to the above criteria, forecasters can employ some promising techniques to further improve the performance of statistical, ML, and DL methods or even combine their strengths. Some of these techniques have already been considered to a noticeable extent in the literature, while others to a smaller, requiring more research. The most notable are summarized below.

- i. **Ensembling:** Combining has long been considered as a useful practice in the forecasting literature and many alternatives have been proposed in order to exploit its full potential. This includes simple operators (e.g. mean, median, and mode of forecasts), trimmed and winsorized averages, information-criteria-based weighting,

error-based combinations (e.g. based on in-sample and out-of-sample forecast errors or ranks), as well as ensembles of forecasts that are produced on processed data (e.g. multiple temporal aggregation, Theta lines, and bootstrapped series), among others (Makridakis et al., 2020). Regardless of the forecasting methods used to provide the base forecasts and the weighting scheme selected for combining them, more often than not, ensembling reduces forecast error and ensures better results in cases where patterns/relationships change and, as a result, the assumptions made by some models may be violated. Especially when it comes to ML and DL methods, which typically involve numerous parameters, initializations, and random sampling, ensembling has proven particularly useful for mitigating issues related to parameter uncertainty, overfitting, and performance stability. As a result, future research should focus on identifying approaches that maximize the benefits of ensembling, examining for instance combinations of ML models that are trained using different initializations, loss functions, multi-step-ahead forecasting strategies, type and number of features, as well as train set sizes. Identifying good practices for forecast combinations and understanding the conditions under which they are expected to perform best is key toward more accurate and robust forecasting solutions.

- ii. **Meta-learning:** Meta-learning refers to the learning process of a model. Effectively, a meta-learner is provided information about the performance of forecasting models of interest and tries to correlate said performance with possible influencing factors. For instance, given some time series features (e.g. strength of trend and seasonality) for a number of series and the corresponding performance of some forecasting methods, a meta-learner could provide useful suggestions about the method(s) that should be preferred for forecasting the same or even different series in the future (Montero-Manso et al., 2020). Similarly, a meta-learner could be used to properly select or combine forecasting methods based on the particularities of the period being forecast (e.g. using as explanatory variables the month of the year, weekday, hour of the day, weather, and special events). Given that meta-learning can improve forecasting performance without introducing new base forecasting models, it has attracted lots of attention recently, finding application in various forecasting tasks. Undoubtedly, popularizing further the

use of meta-learners, introducing more advanced variants of meta-learners, and tackling issues related to additional computational cost<sup>15</sup> could be part of the future work done in the field.

- iii. **Transfer-learning:** A major advantage of global ML and DL models is that they can be used to forecast series that were not part of the original data set used for training them (Tang et al., 2022). This means that once an accurate model is trained, it can be exploited to predict other (similar) series with minimal computational cost and effort (Wellens et al., 2022). Apart from facilitating forecasting at scale, transfer-learning can prove particularly useful for predicting series that lack historical data or as a whole contribute samples that are inadequate in numbers for developing a sophisticated ML or DL model (Alolayan et al., 2022). Although the concept of transfer-learning has been examined with promising results in many forecasting applications, it remains unclear how it should be implemented in practice to obtain the most benefit. For instance, should the pre-trained model be used “as-is” to predict a new set of series or retrained to some extent to better fit the target data set? Also, if the latter is true, how should re-training be applied exactly? For instance, if the pre-trained model is a NN, should we update the weights of the complete set of layers or just the last one, and what kind of hyperparameters should be considered when re-training the model? These are just some of the questions that future research on transfer-learning forecasting could focus on.
- iv. **Data augmentation:** The accuracy improvements achieved by ML and DL methods are usually conditional to the size of the data set available for training. As such, insufficiently rich data sets may limit their utilization and gains. To deal with this issue, data augmentation techniques can be considered to generate synthetic data that resemble and complement the original train set (Bandara et al., 2021). Despite that these techniques are usually inspired by signal and image processing applications, given some proper modifications they can effectively be used for time series forecasting as well. Semenoglou et al. (2023a) reviewed and evaluated some indicative data augmentation techniques, including time series

<sup>15</sup> In order to track the performance of forecasting models under different conditions, extensive simulations (e.g. rolling origin evaluations; Tashman, 2000) are typically required.

flipping, random noise injection, time series combinations, magnitude warping, time series interpolation, bootstrapping, time series generation, and upsampling. Their results supported the use of certain techniques for relatively small data sets. Nevertheless, the successful use cases available and research conducted in the field remain limited (Lainder & Wolfinger, 2022). More importantly, little is known about the properties that data augmentation techniques should have to be proven more beneficial. Should they focus on augmenting rare samples or the most popular ones? Should the technique follow the general tendencies of the data set or be relatively random? Should data augmentation be based on the available samples or aim at generating completely artificial series? Undoubtedly, more work is required to address these questions and popularize data augmentation in time series forecasting settings.

- v. **Advanced DL architectures:** Even though DL has been used effectively for time series forecasting, most of its successful applications mostly rely on deep feedforward and LSTM NNs. On the contrary, more advanced DL architectures widely used for image and text processing, like CNNs and attention-based NNs, remain highly unexplored. As a result, more research would be expected in the following years to identify ways to best modify contemporary DL architectures for time series forecasting.

Regardless of the relative performance of each class of methods, as well as their strengths, weaknesses, and maturity, no one can deny that the field of forecasting has been transformed to a noticeable extent during the last decade, experiencing significant advances in various directions. Arguably, the momentum is strong enough to allow for even greater expectations, pushing forecasters of different backgrounds to work together to further improve forecasting theory and practice and come up with guidelines for selecting the most appropriate method among the statistical, ML, and DL ones. This chapter aimed to provide such guidelines and summarize some practical approaches for achieving better results.

## REFERENCES

- Agarap, A. F. (2018). *Deep learning using rectified linear units (relu)*.  
 Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J.,

- Stella, L., Türkmen, A. C., & Wang, Y. (2019). *Gluonts: Probabilistic time series models in python*.
- Alolayan, O. S., Raymond, S. J., Montgomery, J. B., & Williams, J. R. (2022). Towards better shale gas production forecasting using transfer learning. *Upstream Oil and Gas Technology*, 9, 100072.
- Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, 16(4), 521–530.
- Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., & Petropoulos, F. (2017). Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1), 60–74.
- Bai, S., Kolter, J. Z., & Koltun, V. (2018). *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*.
- Bandara, K., Hewamalage, H., Liu, Y. H., Kang, Y., & Bergmeir, C. (2021). Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognition*, 120, 108148.
- Barker, J. (2020). Machine learning in M4: What makes a good unstructured model? *International Journal of Forecasting*, 36(1), 150–155.
- Bates, J. M., & Granger, C. W. J. (1969). The combination of forecasts. *Journal of the Operational Research Society*, 20(4), 451–468.
- Beaumont, A. N. (2014). Data transforms with exponential smoothing methods of forecasting. *International Journal of Forecasting*, 30(4), 918–927.
- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, T. Hoffman (Eds.), *Advances in neural information processing systems* (Vol. 19). MIT Press.
- Bergmeir, C., Hyndman, R. J., & Benítez, J. M. (2016). Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation. *International Journal of Forecasting*, 32(2), 303–312.
- Bojer, C. S. (2022). Understanding machine learning-based forecasting methods: A decomposition framework and research opportunities. *International Journal of Forecasting*, 38(4), 1555–1561.
- Bojer, C. S., & Meldgaard, J. P. (2021). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2), 587–603.
- Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). *Conditional time series forecasting with convolutional neural networks*.
- Chatigny, P., Wang, S., Patenaude, J. M., & Oreshkin, B. N. (2021). *Neural forecasting at scale*.
- Claeskens, G., Magnus, J. R., Vasnev, A. L., & Wang, W. (2016). The forecast combination puzzle: A simple theoretical explanation. *International Journal of Forecasting*, 32(3), 754–762.

- Crone, S. F., Hibon, M., & Nikolopoulos, K. (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, 27(3), 635–660.
- De Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, 22(3), 443–473.
- De Gooijer, J. G., & Kumar, K. (1992). Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting*, 8(2), 135–156.
- Gilliland, M. (2020). The value added by machine learning approaches in forecasting. *International Journal of Forecasting*, 36(1), 161–166.
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh & M. Titterington (Eds.), *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. Proceedings of machine learning research (Vol. 9, pp. 249–256).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1), 388–427.
- Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., & Yasmeen, F. (2020). forecast: Forecasting functions for time series and linear models. *R package version*, 8, 12.
- Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18(3), 439–454.
- Hyndman, R. J., Ahmed, R. A., Athanasopoulos, G., & Shang, H. L. (2011). Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis*, 55(9), 2579–2589.
- Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*.
- Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1), 167–177.
- Januschowski, T., Wang, Y., Torkkola, K., Erkkilä, T., Hasson, H., & Gasthaus, J. (2022). Forecasting with trees. *International Journal of Forecasting*, 38(4), 1473–1481.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45.

- Kang, Y., Spiliotis, E., Petropoulos, F., Athiniotis, N., Li, F., & Assimakopoulos, V. (2021). Déjà vu: A data-centric forecasting approach through time series cross-similarity. *Journal of Business Research*, 132, 719–731.
- Kang, Y., Cao, W., Petropoulos, F., & Li, F. (2022). Forecast with forecasts: Diversity matters. *European Journal of Operational Research*, 301(1), 180–190.
- Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*.
- Kolassa, S. (2016). Evaluating predictive count data distributions in retail sales forecasting. *International Journal of Forecasting*, 32(3), 788–803.
- Lai, G., Chang, W. C., Yang, Y., & Liu, H. (2017). *Modeling long- and short-term temporal patterns with deep neural networks*.
- Lainder, A. D., & Wolfinger, R. D. (2022). Forecasting with gradient boosted trees: Augmentation, tuning, and cross-validation strategies: Winning solution to the M5 Uncertainty competition. *International Journal of Forecasting*, 38(4), 1426–1433.
- Li, X., Petropoulos, F., & Kang, Y. (2021). *Improving forecasting by subsampling seasonal time series*.
- Ma, S., Fildes, R., & Huang, T. (2016). Demand forecasting with high dimensional data: The case of SKU retail sales forecasting with intra- and inter-category promotional information. *European Journal of Operational Research*, 249(1), 245–257.
- Makridakis, S., Hibon, M., Lusk, E., & Belhadjali, M. (1987). Confidence intervals: An empirical investigation of the series in the M-competition. *International Journal of Forecasting*, 3(3), 489–508.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3), 1–26.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022a). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364.
- Makridakis, S., Spiliotis, E., Assimakopoulos, V., Semenoglou, A. A., Mulder, G., & Nikolopoulos, K. (2022b). Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward. *Journal of the Operational Research Society*, 1–20.
- Miller, D. M., & Williams, D. (2003). Shrinkage estimators of time series seasonal factors and their effect on forecasting accuracy. *International Journal of Forecasting*, 19(4), 669–684.

- Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4), 1632–1653.
- Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1), 86–92.
- Oreshkin, B. N., Carpo, D., Chapados, N., & Bengio, Y. (2019). *N-beats: Neural basis expansion analysis for interpretable time series forecasting*.
- Petropoulos, F., & Spiliotis, E. (2021). The wisdom of the data: Getting the most out of univariate time series forecasting. *Forecasting*, 3(3), 478–497.
- Petropoulos, F., & Svetunkov, I. (2020). A simple combination of univariate models. *International Journal of Forecasting*, 36(1), 110–115.
- Petropoulos, F., Makridakis, S., Assimakopoulos, V., & Nikolopoulos, K. (2014). ‘Horses for Courses’ in demand forecasting. *European Journal of Operational Research*, 237(1), 152–163.
- Petropoulos, F., Hyndman, R. J., & Bergmeir, C. (2018). Exploring the sources of uncertainty: Why does bagging for time series forecasting work? *European Journal of Operational Research*, 268(2), 545–554.
- Petropoulos, F., Grushka-Cockayne, Y., Siemsen, E., & Spiliotis, E. (2021). *Wielding occam’s razor: Fast and frugal retail forecasting*.
- Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Ben Taieb, S., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Cyrino Oliveira, F. L., De Baets, S., Dokumentov, A., ... Ziel, F. (2022a). Forecasting: theory and practice. *International Journal of Forecasting*, 38(3), 705–871.
- Petropoulos, F., Spiliotis, E., Panagiotelis, A. (2022b). Model combinations through revised base rates. *International Journal of Forecasting*.
- Pinson, P., & Kariniotakis, G. (2010). Conditional prediction intervals of wind power generation. *IEEE Transactions on Power Systems*, 25(4), 1845–1856.
- Proietti, T., & Lütkepohl, H. (2013). Does the Box-Cox transformation help in forecasting macroeconomic time series? *International Journal of Forecasting*, 29(1), 88–99.
- Rajapaksha, D., Bergmeir, C., & Hyndman, R. J. (2022). LoMEF: A framework to produce local explanations for global model time series forecasts. *International Journal of Forecasting*.
- Rüse, T., & Tjøstheim, D. (1984). Theory and practice of multivariate arma forecasting. *Journal of Forecasting*, 3(3), 309–317.
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.

- Semenoglou, A. A., Spiliotis, E., Makridakis, S., & Assimakopoulos, V. (2021). Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, 37(3), 1072–1084.
- Semenoglou, A. A., Spiliotis, E., & Assimakopoulos, V. (2023a). Data augmentation for univariate time series forecasting with neural networks. *Pattern Recognition*, 134, 109132.
- Semenoglou, A. A., Spiliotis, E., & Assimakopoulos, V. (2023b). Image-based time series forecasting: A deep convolutional neural network approach. *Neural Networks*, 157, 39–53.
- Shih, S. Y., Sun, F. K., & Lee, Hy. (2019). Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8), 1421–1441.
- Shrestha, D. L., & Solomatine, D. P. (2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2), 225–235.
- Smil, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85.
- Spiliotis, E., Assimakopoulos, V., & Nikolopoulos, K. (2019). Forecasting with a hybrid method utilizing data smoothing, a variation of the Theta method and shrinkage of seasonal factors. *International Journal of Production Economics*, 209, 92–102.
- Spiliotis, E., Assimakopoulos, V., & Makridakis, S. (2020). Generalizing the theta method for automatic forecasting. *European Journal of Operational Research*, 284(2), 550–558.
- Spiliotis, E., Makridakis, S., Kaltsounis, A., & Assimakopoulos, V. (2021). Product sales probabilistic forecasting: An empirical evaluation using the M5 competition data. *International Journal of Production Economics*, 240, 108237.
- Spiliotis, E., Makridakis, S., Semenoglou, A. A., & Assimakopoulos, V. (2022). Comparison of statistical and machine learning methods for daily SKU demand forecasting. *Operational Research*, 22(3), 3037–3061.
- Spithourakis, G. P., Petropoulos, F., Babai, M. Z., Nikolopoulos, K., & Assimakopoulos, V. (2011). Improving the performance of popular supply chain forecasting techniques. *Supply Chain Forum: An International Journal*, 12(4), 16–25.
- Svetunkov, I., Kourentzes, N., & Ord, J. K. (2022). *Complex exponential smoothing*. Naval Research Logistics.
- Tang, Y., Yang, K., Zhang, S., & Zhang, Z. (2022). Photovoltaic power forecasting: A hybrid deep learning model incorporating transfer learning strategy. *Renewable and Sustainable Energy Reviews*, 162, 112473.
- Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, 16(4), 437–450.

- Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013). Regularization of neural networks using dropConnect. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning*. Proceedings of machine learning research (Vol. 28, pp. 1058–1066).
- Wellens, A. P., Udenio, M., & Boute, R. N. (2022). Transfer learning for hierarchical forecasting: Reducing computational efforts of M5 winning methods. *International Journal of Forecasting*, 38(4), 1482–1491.
- Zhang, G., Eddy Patuwo, B., & Hu, Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35–62.



## CHAPTER 4

---

# Machine Learning for New Product Forecasting

*Mohsen Hamoudia and Lawrence Vanston*

## INTRODUCTION

Forecasting the demand for new products is crucial considering the level of investment required for a product launch. (The term “product” here includes services). It is also challenging and risky given global economic competition, evolving customer expectations, and the emergence of new technologies and innovations. Given the high failure rate of new launches (70–80 percent for consumer-packaged goods) (Kosina, 2017), it’s worthwhile to expend significant resources to get the forecast right.

Good forecasts for new products are essential as they help to (a) plan for demand throughout the lifecycle, (b) reduce the risks inherent to any

---

M. Hamoudia (✉)

France Telecom Group (Orange) and PREDICONSULT, Paris, France  
e-mail: [mhamoudia@orange.fr](mailto:mhamoudia@orange.fr)

L. Vanston

TFI, Austin, TX, USA  
e-mail: [lvanston@tfi.com](mailto:lvanston@tfi.com)

product launch, (c) make wise investments and assess potential risks, and (d) adapt manufacturing, planning, and sales processes.

Obviously, new product forecasting is not new. Demand forecasting has been done for decades using judgmental methods, market research like surveys of buyer's intentions, market testing, expert opinion studies like the Delphi method, diffusion models like the Bass model, and statistical modeling through a variety of time series and/or multivariate techniques. Each of these approaches has its strengths and weaknesses, and they can often be used together to create a stronger and more defensible forecast. Our focus here will be on how machine learning (ML) can improve on these traditional approaches.

By definition, we will have little or no history on the demand for a new product. If we wait until the product has been introduced and has begun to be adopted, we could gather enough data points to produce a forecast using time series. The problem with that is many key business decisions must be made *before* there is enough historical data. So, we must do the best with what we have.

ML is a good candidate when we have lots of data, including the sales history, on existing products that are similar to the new one. Although humans use this approach too, the idea is that ML should be able to do it faster and more accurately.

Many papers and case studies are available demonstrating the power of ML in forecasting existing products where there is ample data on their history. For new products, where historical data is limited to similar products, the literature is very limited, but optimistic as to their potential. These two observations (along with our expectation of continued ML progress generally) lead us to our conclusion that the application of ML for new product forecasting will be fruitful in both the research and management spheres.

It is illuminating to briefly review some of the evidence.

Most notably, in the recent M5 forecasting competition (Makridakis & Al, 2022), based on a very large set of retail sales data, ML easily outperformed traditional statistical methods in terms of accuracy and predicting forecast error. The differences were significant only for the top levels. At the bottom levels, minor improvements were observed. The published results do not break out the relative performance of competing methods for existing products vs new products (with short historical data). The winning ML methods took advantage of cross-learning across products and incorporating external data to greatly improve forecasting accuracy

(Makridakis, 2022). This is especially helpful for forecasting new products because it automatically incorporates experience with existing products which we would need to rely on.

The prior M4 competition, which involved only a few microeconomic data and referred to aggregated (company-sector) sales, was won by a hybrid approach of ML and traditional methods, but the benefits of this cross-learning were already becoming apparent. It was subsequently showed that, with cross-learning, ML could outperform both traditional univariate models and the M4 winner (Semenoglou et al., 2021).

The M4 and M5 competitions (Makridakis & Al, 2022) focused globally on existing products, but included also a few products with short historical data. Our understanding is that these results apply somewhat to forecasting new products because it is quite a similar problem. Further, each of the four case studies we review here points to the power of ML in improving the accuracy of demand forecasts for new products.

The rest of this chapter is organized as follows:

Section 4.2 describes the basic categories of new products, along with how they relate to data availability and forecasting. Section 4.3 presents a summary of main forecasting techniques used for new products, focusing on ML. Section 4.4 presents the four case studies of ML and Deep Learning (DL) applied to forecasting new products and Sect. 4.5 provides a summary and conclusions.

## CATEGORIES OF NEW PRODUCTS AND DATA AVAILABILITY

Different forecasting methods apply to different new product categories. For example, it is difficult to apply time series methods for new-to-the-world product forecasts because there is no historical data. We can group new products into four major categories (University of Minnesota, 2016):

- **New-to-the-World Product:** These products are completely new and lead to a fully new market. They comprise 10 to 15% (University of Minnesota, 2016) of the new product category. They are the most challenging to forecast because there is little or no product-specific historical data. Even finding similar products to use as analogies may be a challenge. An example of a new-to-the-world product is the flying taxi, or air taxi, which is a small commercial aircraft that makes short commuter flights on demand.

- **New-to-the-Firm Products:** Also known as new product lines. These products are not new to the world, but haven't been offered by the firm before. These could be simply imitations of products the firm's competitors are already offering or they could represent significant improvements. For example, the French oil company Totalenergie now offers its customers gas, electricity, and renewable energy, each of which was new-to-the-firm products for Totalenergie. For this category, there may be ample historical data on the competition or sector's sales.
- **Additions to Existing Product Lines:** New products in this category are designed to flesh out or extend the product line to serve the firm's current markets. For example, aircraft manufacturer Airbus launched its first jet, the Airbus 300 in 1972, then the Airbus 310 in 1978, then the Airbus 320 in 1984, then the Airbus 330 in 1995, then the Airbus 321 in 1996, and so on. Each new aircraft model met a need for passenger capacity and distance. Similarly with different generations of mobile phones. For this category, experience with previous models may provide useful historical data for new models.
- **Improvements and Revisions to Existing Products:** New products in this category are basically existing products with new or improved features often added to encourage product replacement by existing customers or brand-switching by the competition's customers. For example, smartphone manufacturers like Apple and Samsung constantly launch new models with incremental improvements to great fanfare. For this category, as with the additions category, experience with previous models may provide useful historical data for new models.

### *Data Availability*

For three of the cases, we are likely to have at least some data on similar existing products, but for some new-to-the-world products, which we call *truly* new, we might not. Here, we consider the two cases and what it means for ML-based new product forecasting.

*There is Data on Similar Products that Have Been Adopted in the Past*

We might also have data on some key attributes of these products (e.g., price, size, color, weight, memory, processor, etc., for a new personal computer) and on external variables (e.g., promotions, season, etc.). For example, this would be the case for a large retailer with thousands of product lines. This situation offers several avenues to exploit: (1) similarities in the pattern of adoption across products, (2) the effect of product attributes on demand, and (3) the effect of external variables. Here, ML has proven superior, at least in some respects, to traditional statistical methods (Singh et al., 2019).

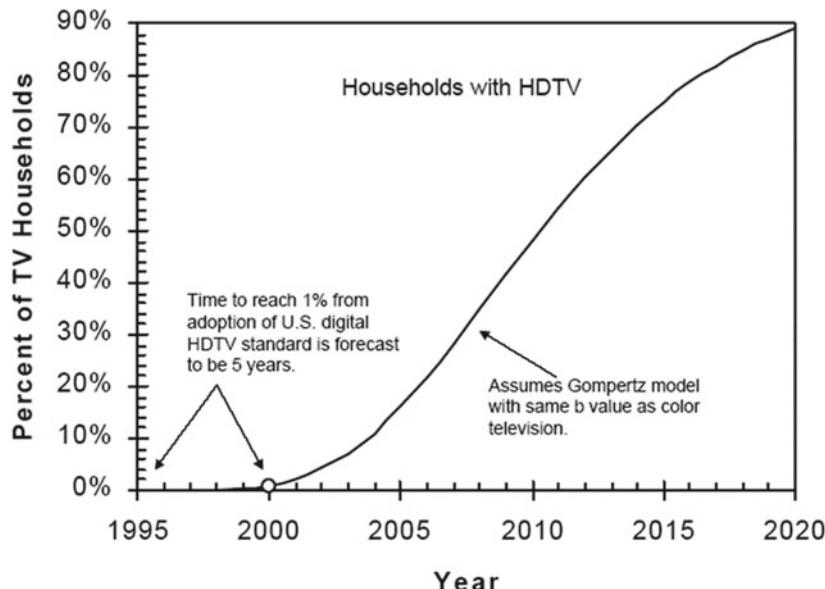
*The Product is Truly New*

We may be able to identify some analogous products, but they aren't close analogies and data on them is limited, qualitative, or difficult to obtain. We may have insights into when or how fast the product will be adopted, but they are not based on "hard" data. We may know the principles of technology adoption, but these are based on years of experience and observation, and, of course, judgment. This is an area where ML, to our knowledge, has yet to be tried.

Figure 4.1 illustrates an example of how a carefully-considered analogy, thorough research, and an analysis of drivers and constraints rendered an outstanding 1995 forecast of high-definition television adoption in the US (Vanston et al., 1995). This new product forecast required judgment about (a) when the new product would begin to be adopted in significant quantities (1% in 2000), whether it would be successful (yes), ultimate market penetration (~100% of households), appropriate model (Gompertz, typical of consumer adoptions), and rate of adoption (same as color TV, 1955–1992) for which each was highly controversial at the time (Vanston, 2008).

## HIGHLIGHTS OF THE MAIN FORECASTING TECHNIQUES USED FOR NEW PRODUCTS

Demand forecasting for new products and services has been done for decades using many categories of forecasting techniques (Meade, 1984). Each has its strengths and weaknesses depending on the forecast objectives and the structure of the historical data. They can often be used together to create a stronger and more accurate forecast (Ord et al., 2017).



**Fig. 4.1** 1995 High-definition television forecast (Vanston et al., 1995) (*Source* Technology Futures, Inc.)

### *“Traditional” Methods*

“Traditional” methods highlighted in Figure 4.2 (Hamoudia, 2021) span both qualitative (judgment and market research) and quantitative (time series, regression, and cause-effect) forecasting. In their article “The evolution of forecasting techniques: Traditional versus machine learning methods,” Menon S., Ranjan R. (2020) wrote: *“In general, traditional algorithms tend to use predefined techniques and statistical models such as linear regression, autoregressive integrated moving average (ARIMA), and autoregressive integrated moving average with explanatory variable (ARIMAX). The goal of traditional forecasting methods is largely descriptive in nature, aimed at analyzing a univariate dataset or a multivariate dataset with finite, countable, and explainable predictors.”*

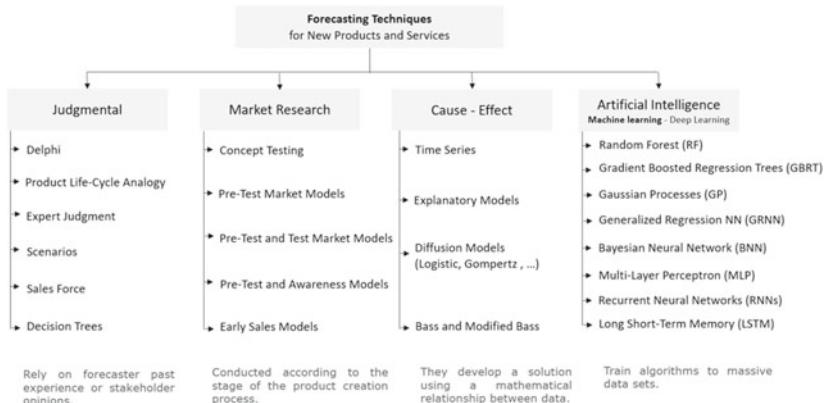


Fig. 4.2 Forecasting techniques for new products (Hamoudia, 2021)

### *Expert Opinion*

Expert opinion, which falls under both categories of judgmental and market research, can be deployed in numerous ways (Gilliland et al., 2015). For example, experts can forecast when a product will be introduced, what the total potential market is, and when various levels of market penetration will be reached, including market saturation. Any number of expert methods can be employed ranging from single interviews or simple surveys to focus groups and Delphi studies. Expert opinion can include existing forecasts, projections, and plans—maybe a short-term plan here and a long-term target there, for example.

Group forecasting exercises are an excellent way of leveraging expert opinion and are easy to implement and relatively economical. Gathering the opinion of participants from the various functions of the organization (e.g., marketing, production, sales, and R&D) helps reach consensus on the forecast and avoids surprises. This can be repeated periodically to update the forecast. External consultants and experts can be invited to participate and provide information and perspectives (say, on competition and regulation) that might otherwise be missing.

### *The Delphi Method*

Delphi is an expert opinion method that is well suited to long-term forecasts in a market or industry that expects significant changes or external events that will impact sales (Armstrong, 2004). A group of experts with

knowledge on the subject matter is organized. Each expert individually and anonymously responds to a survey where each makes quantitative assessments (e.g., likelihood of success) and forecasts (e.g., total demand by a given year) regarding specific items (new products, in this case) along with their reasoning. There is little chance that a consensus on the forecast will be reached as early as this first round. We then move on to a second stage where these forecasts are aggregated by an external Delphi coordinator and shared with the group, with answers always kept anonymous. The same group then completes the survey again and has the option to modify its responses based on the previous answers. This sometimes requires three to four rounds before a final consensus on the forecast is reached.

#### *Consensus of Sales Team*

This is an expert opinion method that integrates the business and sales people of the company who are close to current customers and in contact with prospects (Makridakis et al., 1983). Each person can give an opinion on sales forecasts and how customers will likely react to a different product or service. As there are often issues in terms of trade objectives, participants may be overly optimistic or pessimistic in their estimates. The answers can then be averaged, for example, to make the final forecasts.

#### *Customers Surveys*

These explore customer expectations for a new product, for example. Customer surveys can help identify key trends and changes in the market, leading to sales forecasts (Gilliland et al., 2015).

One of the formal methods in this category is conjoint analysis (Hielskrem, 2008). Conjoint analysis is a survey-based statistical technique for measuring the value that consumers place on features of a product or service. It consists of creating, distributing, and analyzing surveys among customers to model their purchasing decision based on response analysis. It combines real-life scenarios and statistical techniques with the modeling of actual market decisions. It is useful for both new product launches and repackaging existing products. Conjoint analysis is considered by many analysts as the best survey method for determining customer values.

### *Drivers and Constraints*

In a nutshell, *Drivers and Constraints* (Vanston, 2008) attempts to answer the following questions:

- What are the drivers for adoption? How strong are they?
- What are the constraints on adoption? How strong are they? Can they be overcome?
- What is the balance of drivers and constraints? Will this change and, if so, when?
- What are the important areas of uncertainty that need to be resolved? How can these be answered?

If the scale tips strongly against the product, its chances are marginal at best. If the scale tips strongly in favor of the technology, it estimates optimistic forecast parameters. Most often, there is a rough balance and the key issue is whether and when the constraints will be overcome and/or when the balance of drivers and constraints change. Usually, there will be one or two key areas of uncertainty or controversy and much effort will have to be devoted to addressing those areas.

### *Diffusion Models (S-Shaped Curves)*

Diffusion models are based on the idea that product adoption (the percentage of the market that has adopted the product or its successors) usually follows an S-shaped curve, of which there are numerous models to choose from (Meade, 1984). The commonly-used models have only a few parameters. These models track market penetration, not sales, although first time sales can be derived from the rate of change.

The simplest models are the logistic and Gompertz curves (Sokele, 2008). These include a parameter that controls the rate of adoption and one that places the adoption in time. A third parameter for the market saturation level can be added, but often the data is just scaled to a presumed market saturation, so the parameter is set to one. Of the two, the logistic model is most applicable to industrial and commercial adoptions, while the Gompertz model is more applicable to consumer adoptions (Vanston, 2004).

The Bass model (Bass, 1969) is slightly more complex and is very popular among market researchers. It distinguishes between the adoption

rates of innovators (early adopters) and imitators. Adding another parameter, it also increases the flexibility of the S-shape curve (i.e., more curve shapes can be represented). A number of modified forms have been developed, including the Generalized Bass model which includes covariates (Bass et al., 1994).

There are also a number of variations on the basic models for handling multiple substitutions, differentiated market segments, capital, or other constraints (Wright & Stern, 2015).

Whatever the model, a small number of parameters must be estimated to make a forecast. If there is sufficient historical data, the parameters can be estimated using regression (Lee et al., 2014). However, as mentioned already, this is not the case for new products. The most common method for estimating the parameters is finding analogous partial or completed product adoptions, estimating their parameters via regression, and assuming the new product's parameters will be similar (Vanston, 2004).

Some new products require improvement in price and/or performance to be widely adopted. (Moore's Law is a good example of a performance trend). Since there are proven methods for forecasting these types of trends, we can use them to help forecast the adoption curve, especially if we understand the relationship between the two.

### *Machine Learning Methods for Predicting New Product Demand*

Artificial Neural Networks (ANN) and Gradient Boosted Trees (GBT) are the ML approaches that have been most applied to multivariate forecasting problems. Random Forest, an older tree-based ML approach, is still used as well (e.g., in one of our case studies).

ANNs generate the most excitement and dominate research, while GBTs dominate commercial applications and recent contests such as the M5 competition. Although the approaches are competitive in terms of accuracy, the GBTs tend to be easier to use, more robust, stable, less cranky, more intuitive, easier to interpret, deal well with missing data, and are nicely packaged in popular software. ANNs require more expertise, finesse, and skills in tweaking the general architecture to meet specialized forecasting needs (Januschowski, 2022).

For academics and well-staffed, research-oriented groups in industry, ANNs are still the main focus and, perhaps, have more long-term promise. However, for the practitioner wanting to get a new product forecasting

system operational quickly, GBTs are the logical choice. However, we will review both approaches and explain how they work and what the challenges present are.

### *Gradient Boosted Trees (GBT)*

GBT algorithms employ groups of decision trees optimized using gradient boosting. They are very popular for demand forecasting. They come in several flavors including XGBoost (XGB), LightGBM (LGBM), and CatBoost, the latter designed for categorical data. Boosting is a ML ensemble technique where new models (decision trees in the case of GBT) are added to correct the errors made by existing models. Models are added sequentially to try and correct for errors in prior models until a stopping condition is satisfied. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

XGB and LightGBM are very similar, but each has its adherents and its relative advantages which may be case specific. LightGBM is newer and often described as faster and more accurate, but, again, not always. For our two case studies that compared the two, XBG was more accurate in one and the LightGBM, in the other. However, most of the M5 competition winners used LightGBM. Makridakis et al. (2022) suggest that LightGBM dominated the M5 competition because: “*... it allows the effective handling of multiple features (e.g., past sales and exogenous/explanatory variables) of various types (numeric, binary, and categorical). In addition, it is fast to compute compared with other gradient boosting methods (GBMs) [in this case, GBTs], does not depend on data pre-processing and transformations, and only requires the optimization of a relatively small number of parameters (e.g., learning rate, number of iterations, maximum number of bins, number of estimators, and loss functions). In this regard, LightGBM is highly convenient for experimenting and developing solutions that can be accurately generalized to a large number of series with cross-correlations.*” They also note that LightGBM dominated other recent forecasting competitions as well. The observation about hyperparameters is important because their proper selection can be crucial to forecast accuracy.

Although the above commentary concerns forecasting existing products, given the similarities, LightGBM is probably the best place to start when developing a system for forecasting new products.

### *Artificial Neural Network (ANN)*

Artificial Neural Networks (ANN) are similar, in principle, to the networks of neurons in our brains. In either case, information is passed along the network by the strength of the connections among the neurons. And, in either case, the network learns from data it receives and provides responses in the form of quantitative predictions (regression) or classifications. In the case of ANNs, the network is organized in layers of neurons with an input layer, one or more hidden layers, and a final output layer. Early ANNs tended to have a single hidden layer, while now ANNs tend to have at least several hidden layers allowing what is called *deep learning*.

Technically speaking, ANNs are nonlinear statistical models capable of capturing very complex relationships between the inputs and outputs, much more so than, say, traditional multiple regression. They are also great at recognizing patterns, making them suitable for a variety of tasks, such as image recognition, speech recognition, machine translation, and medical diagnosis. ANNs are also good at function approximation, providing a cost-effective method of arriving at the solutions that define the distribution of the examined data. With ANNs, one can enhance existing data analysis techniques owing to their advanced predictive capabilities (Jeffrey et al., 2000).

Over time, a number of specialized ANNs have been developed. For example, Convolutional Neural Networks (CNN) are particularly suited to pattern recognition, especially in images. Recurrent Neural Networks (RNN), including the Long Short-Term Memory (LSTM) version, have feedback loops and memory which make them ideal for processing sequential data of the type encountered in forecasting. Benidis et al. (2022) provide more details on the application of ANNs for forecasting.

More recently, transformer networks have emerged as highly-efficient, highly-scalable ANNs capable of handling massive amounts of sequential data. To date, they have mostly been applied to large language models such as ChatGPT, but they are beginning to be used for forecasting as well. In a recent study (Kunz et al., 2023) using a large data set from an online fashion retailer, a transformer model was tested against a naive forecast that simply repeats the last observation, LightGBM, and an autoregressive RNN (DeepAR). The authors conclude: “...our transformer model performs best in terms of demand error and MSE. While tree based models [i.e., lightGBM] seem to be a close competitor in terms of demand error and were much faster to train, we saw a significantly higher bias. In terms of demand bias and MAPE, both Deep Learning models are

*very close to each other, but for RMSE again lightgbm and our transformer model have a clear advantage.*" The literature has been unclear whether the scaling laws (basically stating that accuracy keeps improving as we add more data) proven in large language models apply to forecasting. In an experiment on the data, the authors found evidence that they indeed do, although more research is required.

As powerful as they are, using ANNs is not always easy. They require masses of data, lots of computation time, significant expertise, and many hyperparameters (i.e., variables that control the operational details of the ANN algorithm) that must be set correctly. In a recent study, (Semenoglou et al., 2023), the authors assess that using ANNs introduces new challenges for forecasters, mainly because of their stochastic nature and the large number of hyperparameters influencing their performance. The results of their study suggest that ensembling ANNs significantly boosts forecasting performance, but at the cost of additional computational time.

Although we recommend GBTs as the place to start because of their ease of use, robustness, and accuracy, ANNs continue to be used by experts in research and in practice, and the generality of ANNs (like our brains) allow for the development of highly-specialized models capable of deep learning. And with the introduction of transformer models for demand forecasting, there is promise for their further progress, including, potentially, for new product forecasting.

### *Structuring the New Product Forecasting Problem*

Apart from selecting an ML algorithm, there is the issue of how to structure the forecasting problem. For example, we know we are going to rely on sales data on existing products that are similar to the new product. We will likely have a number of attributes that characterize the products (like price, type, color, and size) that we believe may be important in determining demand. But how do we determine which products are similar? Do we pre-define classes (groups of products that are similar) and label each product with its class (say, expensive suits for tall men)? Or do we let ML create the classes for us, using unsupervised clustering algorithms like K-means? Or do we just feed the attributes to the ANN or GBT as features and let the algorithm learn to forecast demand without explicitly identifying classes? Our case studies explain these options in more detail.

Another issue is how broad a net to cast. Do we include lots of products, even though we think they are dissimilar to the new product and let

the algorithms decide whether they are or not? Sales data from a dissimilar product may have relevant information (maybe expensive car sales illuminate expensive suit sales) or maybe the information is irrelevant or misleading.

Still a third decision involves whether to divide the forecasting problem into parts. For example, do we forecast market size (say total sales) and the pattern of demand by period separately? Or do we let the algorithm directly predict demand by period?

Interestingly, our four case studies cover all of these options! There are arguments pro and con for each choice and the choice may be case specific. It is too early to tell if one choice is universally better for forecasting new products.

### *Potential Machine Learning Approaches for Truly New-to-the-World Products*

All of the ML approaches to forecasting to date have involved large data sets to train on. This brings us back to our earlier point about truly new-to-the-world products, where such databases may not be readily available. For these products, which often have a forecasting horizon of many years, forecasting may continue to be done the traditional way for the time being.

A typical traditional approach is to use a diffusion model with the parameters estimated based on analogies to hand-selected existing products. This sounds very similar to what ML is doing automatically in the case studies reviewed below, as well as in winning models in the M5 competition. One question to be determined is, for truly new products, will ML estimate the S-shape curve parameters, like humans do, or will predict the sequence of future values without reference to any functional form? At first glance, one would expect the latter, but it could go the other way. The S-shaped curves contain much experience and theory. Given the dearth of relevant analogies, there will be less data to crunch and so fewer degrees of freedom may be advantageous.

As of now, it is unclear if ML will do (or should do) other things that have proven effective in traditional forecasting, like identifying and weighing drivers and constraints, incorporating expert opinion, reviewing a range of literature for relevant information, and assessing supporting trends. ML is doing some of these things in other domains, for example, large language models, so it appears to be possible. More research is

required, but given enough data and more powerful algorithms, ML may be able to tackle the forecasting of truly new-to-the world products with the same creativity—and accuracy—as humans.

## A REVIEW OF FOUR CASE STUDIES OF MACHINE LEARNING FOR NEW PRODUCT FORECASTING

ML has proven superior in many respects to traditional statistical methods for forecasting existing products when large multivariate datasets are available. In this section, we review four recent case studies that indicate the same is true for new product forecasting *if* there is enough data on similar existing products.

### *Forecasting Demand Profiles of New Products*

Van Steenbergen, R. M., Mes, M. R. K. (2020)

Van Steenbergen, Mes, and M. R. K. (2020) developed and tested a method called DemandForest to make prelaunch forecasts and support inventory management decisions for new products using a combination of K-means, Random Forest, and Quantile Regression Forest. The method is based on historical sales data of previously introduced products and the product characteristics of existing and new products. They broke the problem into two parts: (1) forecasting total demand for the new product over the forecast horizon and (2) forecasting the pattern of that demand over time, that is, the percentage of total demand in each period. Multiplying the two parts yields the new product forecast.

#### *Description*

Total demand was forecasted using Quantile Regression Forest (QRF). As described by the authors, “QRF grows a set of trees in the same way as Random Forest. However, instead of keeping only the mean of the observations, QRF keeps all observations for each leaf node in each tree. Using all observations, QRF approximates the full conditional distribution and can provide a prediction interval for new observations.” This way it can quantify uncertainty, which is important in inventory management. The model was trained on the total sales (over the same number of periods as the forecast horizon) and the characteristics of the existing products.

Then, the characteristics of the new product were applied to the trained model to forecast total demand along with a prediction interval.

The demand pattern was forecasted by first using K-means clustering applied to the historical sales data to identify representative sales patterns over time. Once these patterns were identified, a Random Forest was trained on the demand patterns and the characteristics of the existing products. Then, the characteristics of the new product were applied to the trained model to forecast the pattern for the new product, which is then multiplied by the total demand to yield the demand forecast.

### *Test Setup*

The authors tested the DemandForest method on five real data sets from industry and one synthetic data set generated from an idealized model. The real data sets were from five different companies spanning a range of industry and product types as Table 4.1 summarizes. For each product, the first 18 weeks of sales after introduction were selected, so the forecast horizon was 18 weeks. The training set consisted of a random sample of 75% of the data, while the test set consisted of the remaining 25%. The training set essentially represents the existing products, while the test set represents the new products to be forecast.

The K-means algorithm was run 25 times for each data set. For the five real data sets they examined, two optimal clusters emerged: one with relatively constant sales and the other with declining sales. The synthetic data set resulted in three optimal clusters. The Random Forest and QRF algorithms were run with 2000 trees.

The model was evaluated on how well the forecasts match the actual data for the test set. For each data set, the authors reported the Root Mean Square Error (RMSE) for DemandForest, two variations (that fit continuous Gamma and Log-Normal distributions to the empirical distribution in the QRF), and two benchmarks. The first benchmark, abbreviated ZeroR, is the average demand for each period of the training data. The second, called OneP, attempts to mimic human judgment by finding the one product in the training set that most closely matches the characteristics of the new product. The total demand for that product is applied to the average demand profile in the training set to determine the OneP forecast for each period.

**Table 4.1** Brief description of the companies and new products (Steenbergen & Mes, 2020)

<i>Company</i>	<i>Industry type</i>	<i>Market</i>	<i>Type of products</i>	<i>Product characteristics</i>	<i>Number of Products</i>
A	Retail	B2C	Household items	Supplier, category, sales price, margin, collection type, product group, space facing, circle type	16,229
B	E-commerce	B2B & B2C	Lighting	Supplier, category, purchase price, sales channels, article type	3197
C		B2C	Sanitary ware	Supplier, category, subcategory, subsubcategory, sales price, margin, product type, brand, brand collection	592
D	Wholesale	B2B	Agricultural machinery (spare) parts	Supplier, category, purchase price, brand	1172
E	Wholesale	B2B	Garden tools and forestry machines	Supplier, category, purchase price, brand	660

## Results

Table 4.2 summarizes the results. For the synthetic data and data sets A, B, and E (household items, lighting, and garden tools & forestry machines), DemandForest was more accurate than either benchmark. For data sets C and D (sanitary ware and agricultural machinery spare parts), there was little or no improvement over the best benchmark. (For data set C, DemandForest outperformed the benchmarks on total demand, but fell short on the demand profile).

The authors also tested for the accuracy of the 90% demand intervals. The accuracy of the total demand intervals forecast by the QRF was good in all cases, and sometimes better than the benchmark intervals. However, for data sets A and B (as well as the synthetic one), the intervals of the by-period forecasts by DemandForest were too narrow, under-performing the best benchmark. They were about right for data

**Table 4.2** Predictive performance of DemandForest compared to benchmarks (Steenbergen & Mes, 2020)

<i>Method</i>	<i>Synthetic</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
DemandForest	10,8	0,681	32,5	2,22	3,59	3,61
DF + Gamma	10,8	<b>0,676</b>	<b>32,0</b>	2,15	3,48	3,56
DF + Log-Normal	<b>10,7</b>	0,683	32,6	1,83	<b>3,03</b>	<b>3,31</b>
Benchmarks:						
OneP	13,2	0,705	40,2	2,57	4,11	5,36
ZeroR	15,2	0,784	36,4	<b>1,78</b>	3,15	4,24

sets C and E, but so were the benchmarks. For data set D, both DemandForest and the benchmarks forecast too wide an interval. So, for interval accuracy, DemandForest was okay, but needs some improvement.

#### *Feature Importance and Comparable Products*

Apart from the forecasts, the authors also used the Random Forest to derive, for a given data set, the relative importance of the product characteristics in determining demand (permutation feature importance), and the top five most comparable products (based on the proximity matrix).

#### *Summary and Conclusion*

In summary, the DemandForest approach can improve forecasting for new products, sometimes significantly, as well as providing insight to planners. When it stumbles, the problem appears to be in the demand profile selection, an area where we would hope to see improvement with additional relevant characteristics (the authors suggest seasonality, weather, and internet traffic).

#### *Fashion Retail: Forecasting Demand for New Items*

Singh P. K., Gupta Y., Jha N., Rajan A. (2019):

As described by the authors, and by Luellen (2020), forecasting demand for fashion items is complex and difficult due to “the transient nature of trends in colors, prints, cuts, patterns, and materials in fashion, the economies of scale achievable only in bulk production, as well as geographical variations in consumption.” Analysts, statisticians, and ML researchers have tested many models that forecast future demand for a

particular item of fashion with historical data on its sales. The authors write that, to their knowledge, “none of the tested models have generalized well to predict future demand at an abstracted level for a new design/style of fashion article.”

To address this problem, the authors present “a study of large scale fashion sales data and directly infer which clothing/footwear attributes and merchandising factors drove demand for those items.” They then built “generalized models to forecast demand given new item attributes, and demonstrate robust performance by experimenting with different neural architectures, ML methods, and loss functions.

### *Methodology*

The authors included in the model merchandising variables such as discount, visibility, and promotion. *Discount* was measured as the deviation from a brand’s average discount and the average discount on the retail platform overall. *Visibility* was measured by views of the page on which the apparel item was listed, relative to the average views for the brand and overall platform views. *Promotion* was measured by the days until or after a planned promotion.

Also included were derived factors affecting sales: age of the style, trend, seasonality, and cannibalization. *Age of the style* was derived by calculating the number of days the item had been available for sale. *Trend* (in consumer interest over time) was calculated by the number of weeks between the “experiment start date” and the current date. For *seasonality*, the first three terms of a Fourier transformation of week of the year were used as the features. For *cannibalization*, a calculation was made relating to the number of similarly styled items available in one brand in a given week versus other brands, and similar styles of the same price range.

The authors built a model that learns to identify similarly behaving time series across latent parameters, and also account for discounting, promotions, visibility variations in comparing the time series. A point in a time series is represented as:

$$y_{it} = f(A_i, M_{i,t}, M_{i,t-1}, \dots, M_{i,t-p}, D_{i,t}, D_{i,t-1}, \dots, D_{i,t-p})$$

where  $y_{it}$  is sales for item  $i$  at time  $t$ ,  $A_i$  are the attributes of item  $i$  like color (e.g., blue), material (e.g., cotton), etc.,  $M_{i,t}$  are the merchandising factors like discount and promotion for the item  $i$  at time  $t$ ,  $D_{i,t}$  are the derived features like trend and seasonality, and  $p$  is the time lag.

In the data, the authors noticed the long tail, characteristic of retail, where a minority of items account for the majority of sales. This results in a variation in sales of more than several orders of magnitude. To address this high variance problem, the authors trained the models using both log and linear scales and tried different loss functions.

The following algorithms and alternative loss functions were compared:

1. Model: Random Forest (RF) - Loss function: Mean Squared Error (MSE)
2. Gradient Boosted Regression Trees (GBRT) - Loss function: MSE and Huber
3. Light Gradient Machine (LGBM) - Loss function: MSE and Poisson
4. CatBoost (CB) - Loss function: MSE and Poisson
5. XGBoost (XGB) - Loss function: MSE and Poisson
6. Attribute Embedding + MLP - Loss function: MSE and Poisson
7. Attribute Embedding + LSTM - Loss function: MSE and Poisson

The tree-based models and MLP were trained in a nonlinear ARIMA manner, where lagged values of time varying features are used to capture temporal dependencies.

### *Experiment and Results*

The authors tested the models on five different data sets, each for a different product, namely shirts, casual shoes, kurtas, tops, and t-shirts, with the number of items in the tens of thousands in each set. The average weekly sales for each style served as a naïve benchmark. Accuracy was compared using wMAPE. All of the top five models significantly outperformed the naive benchmark across the product categories. The authors concluded that, among the selected models, XGB consistently performed the best, or near best, to predict demand for new fashion trends.

### *Conclusion and Future Directions*

The authors conclude:

Contrary to our initial expectations, DNN [Deep Neural Network] models (LSTM and MLP) did not show better performance over tree based models. LSTM seemed like a good choice of model theoretically since it has been shown to perform very well over various time series data,

and is architecturally better suited to model long temporal dependencies. We intend to explore further in this direction by building an appropriate RNN architecture for demand forecasting that generalizes across datasets of different article types in fashion without overfitting. We are also experimenting by including image based features in our forecasting models along with currently used textual attribute embeddings.

### *New Product Forecasting Using Deep Learning - A Unique Way*

Affine, Post of August 2022, on <https://affine.medium.com/new-product-forecasting-using-deep-learning-a-unique-way-73ebdf628dbc>

In *New Product Forecasting using Deep Learning: A Unique Way* (2022), a post published in August 2022, the author employs a two-part approach using Neural Networks to forecast demand for newly-launched footwear styles for an apparel retailer.

It is important to highlight that this post is not really an “article” in the academic sense as it has not been peer-reviewed. In addition, we do not know if the out of sample analysis has been done properly.

#### *Methodology and Results*

In the first part, a Convolutional Neural Network (CNN) is trained on a database of existing shoe images to classify images of new styles according to product attributes (brand, color, heel height, style, etc.). As part of this step, the existing shoes that best match the new style are identified using similarity indices based on the normalized overall probability scores.

In the second, a Recurrent Neural Network (RNN) is trained on sales data from existing products, along with factors such as promotions, pricing changes, seasonality, average customer ratings, and the product attributes from the CNN. The RNN implementation was done using the Keras Sequential model and the loss function was estimated using “mean squared error.” The author does not report the number of products or observations.

The author determines that the accuracy of the CNN/RNN method is in the range of 85–90% compared to 60–65% for the existing method used by the retailer, which involved heuristics for selecting analogous products and undescribed “supervised techniques.”

### *Summary and Conclusion*

Based on 39 products, the author has shown that using deep learning significantly improves forecast accuracy over the existing methodology (from 60–65% to 85–90%). In addition, the author notes that the overall process can be scaled quickly since it is self-learning and that forecasts can be made faster and more efficiently due to the automation of decision-intensive processes like the selection of analogous products. The author warns, however, that image-matching training requires a huge amount of data and that managing image data is a costly.

The author also sees opportunities to improve the process. For example, “the feature selection method can be automated through unsupervised techniques like Deep Auto Encoders which will further improve scalability” and they believe accuracy can be improved further with a deeper network and an additional one-time investment in processing.

### *Forecasting New Product Demand Using Machine Learning*

Smirnov P. S., Sudakov V. A. (2021)

Smirnov P. S. and Sudakov V. A. (2021) used GBT to forecast demand for new products based on extensive data from the Russian e-commerce company Ozon. Across all product categories, there were data for about 89,000 items with sales from as early as 2012 to 2020. The model considers product attributes such as price, promotion, and product category (24 categories such as sports, furniture, and books) and a set of features based on historical sales data like year, day of year, day of the week, first week’s sales, etc. Their general objective was “to propose a forecasting method based on ML models to forecast the demand of new products satisfying the following conditions: (a) does not depend on the type of goods, (b) can work without history of sales, (c) can work with large data set, and (d) no need for marketing research.”

### *Methods*

The authors tried three GBT varieties—XGBoost, LightGBM, and CatBoost. They state that gradient boosting for regressions is a “powerful technique for combining multiple ‘base’ classifiers to produce a form of committee whose performance can be significantly better than that of any of the base classifiers (Bishop, 2006)” listing the advantages of boosting trees as natural handling of “mixed” type data, handling of missing values,

robustness to outliers in the input space, computational scalability, and high accuracy. Due to its computational efficiency, they used the gradient descent learning algorithm (Hastie et al., 2001).

### *Results*

Among the three varieties of GBT (XGBoost, LightGBM, and CatBoost), the authors concluded that LightGBM achieved the best results in their case. Random search was used to optimize the hyperparameters. Accuracy is reported in terms of the RMSE during training and testing/validation, achieving a RSME of 4.0. The authors do not provide a benchmark, noting that one does not exist for their data set. That said, the starting solution for the GBT models, which had a RSME of 5.0, serves as a rough naïve benchmark. The authors note, “Firstly, despite the complex data, the error decreases during the learning process, which indicates that the algorithm is being trained correctly. Secondly, no increase is observed in the error on the validation set, which means that the effect of overfitting was successfully avoided, and the model will work correctly on new data.”

### *Summary and Conclusion*

The authors wrote that “the proposed model satisfies initial conditions because it meets all the requirements – forecasting demand without any marketing research and to work independent of the type of goods and historical data.” They add that the “community can use this model for predicting software development which could successfully work with very complex problems of predicting new good demand.”

The authors also ranked feature importance based on the amount the prediction changes if the feature value is changed. Day of the year, price, and discount were determined to be most important. Interestingly, the product category was eighth most important, indicating that the model might be quite general across products. They don’t offer any benchmarks to compare their accuracy, so further evaluation of the model is difficult.

They plan in future studies “to improve accuracy by using features extracted from other types of data like images and videos.”

### *Summary and Lessons from the Four Case Studies*

Although they vary widely in approach, all four studies confirmed the value of ML in forecasting new product sales. The studies differed in

how they measured the accuracy improvement afforded by ML including comparing the ML predictive accuracy to naïve, yet effective, benchmarks, a benchmark intended to simulate human judgment, the “existing methodology,” and simply the starting solution. The lack of a common benchmark is understandable, since there are not common benchmarks based on standard statistical methods for forecasting new product demand like there is for existing products. Needless to say, standard benchmarks of some type for new product forecasting would be a welcome development.

A very relevant benchmark is what humans can do. If ML can do as well or better without the expense, effort, and time required for a human-driven project, then ML will likely find its place in new product forecasting. Of course, the startup cost of building an ML system—data acquisition and management, computation for training and testing, labor for system design and implementation—can be large, so that investment will need to be justified by the system’s long-term benefits.

The value of ML seems robust across a wide variety of industry types. Two of the studies were limited to the retail fashion industry, but one study included two dozen widely-varying product categories on an e-commerce platform, and another study included data from five different product types spread across wholesale, retail, and e-commerce.

One of the studies used Random Forest, one used only GBT models, one used only ANN models, and another tried all three, concluding that, for their data set, the GBT generally performed better. There is not enough evidence in the case studies to conclude that GBT is more accurate than ANN or Random Forest for new product forecasting, especially since there’s no common data set or metric across the studies. However, the M5 leaderboard was dominated by GBT approaches, and while the M5 was about existing products, it likely has significance for new product forecasting as well. So, at the very least, GBT may be competitive in accuracy. And it is currently preferable in terms of robustness, interpretability, and ease of use. This makes it the ML method of choice for most managers seeking to implement a system new product forecasting. As noted already, LightGBM appears to be the best choice among the GBT methods for demand forecasting.

However, in research, ANNs are still of great interest and continue to fascinate. With improvements in accuracy and the other factors, by the time you (human or machine) read this, the tables may have been turned.

Three of the studies reported using data sets with 20,000 or more products (one didn’t report the number). This includes the Steenbergen

and Mes data set, although they analyzed each of their five companies individually with the number of products per company being as low as 592. Although we don't have enough information to recommend a minimum, managers can count on needing a significant data base of similar products to use current ML technology effectively.

## SUMMARY AND CONCLUSIONS

Forecasting the demand for new products is risky and crucial given the level of investment required for a launch. Accuracy of demand forecasts of new products is a top priority for decision-makers. Underpredicting demand leads to a loss of potential sales; overpredicting it leads to costly excess inventory. Decision makers tend to believe the forecast that is based on the consensus of forecasts from two or more forecasting methods. Each of the various approaches has its strengths and weaknesses, and they can often be used together to create a stronger and more defensible forecast. ML enriches the various existing traditional methods and can improve the accuracy of our forecasts. And perhaps in the future, the various traditional methods can help ML become more accurate for difficult long-term forecasts for truly new-to-the-world products.

The application of ML to forecasting new product is in its infancy. We have made some recommendations, but each ML algorithm has its strengths and weaknesses, so it is crucial to choose the right algorithm that will work best for the specific data set and problem. To achieve this, it is important to (a) understand the project goal, (b) analyze the data by size (should it be prepared and structured for training process?), (c) evaluate the speed and training time, and (d) decide on the number of features and parameters. Further, the ML model should be updated and monitored regularly to reflect changes in the data set of similar and analogous products used to forecast the new product, not to mention the inevitable advances in ML down the road. Ultimately, it is important to emphasize to perform good benchmarks and widespread testing.

**Acknowledgements** We would like to thank Evangelos Spilios (National Technical University of Athens, Greece) and Robert Fildes (Centre for Marketing Analytics and Forecasting Lancaster University Management School, United Kingdom) for reviewing our article and Robert Van Steenbergen for allowing us use two tables contained in this article (R. M. Van Steenbergen, M. R. K. Mes, 2020).

## REFERENCES

- Armstrong, J. S. (2004). *Principles of forecasting—A handbook for researchers and practitioners*. Kluwer Academic Publishers.
- Bass, F. M. (1969). A new product growth for model consumer durables. *Management Science*, 15, 215–227.
- Bass, F. M., Trichy, V. K., & Dipak, C. J. (1994, Summer). Why the bass model fits without decision variables. *Marketing Science*, 13(3), 203–223.
- Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., & Aubet, F. X. (2022). Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6), 1–36.
- Bishop, C. M. (2006). *Pattern recognition and machine learning* (p. 657). Springer.
- Gilliland, M., Tashman, L., & Sglavo, U. (2015). *Business Forecasting*, J. Wiley & Sons, Inc.
- Hamoudia M. (2021). Forecasting new products using statistical methods vs machine learning, IIF Workshop, Madrid, Spain, 9 December 2021.
- Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction* (p. 361). Springer.
- Hielkrem, C. (2008). Market scenarios and conjoint analysis, in Telektronikk, 3 / 4.08.
- Januschowski, T., Wang, Y., Torkkola, K., Erkkilä, T., Hasson, H., & Gasthaus, J. (2022). Forecasting with trees. *International Journal of Forecasting*, 38(2022), 1473–1481.
- Jeffrey, R., Song, T. M., & Calantone, R. J. (2000, November). Artificial neural network decision support systems for new product development project selection. *Journal of Marketing Research*, 37(4), 499–507 (9 pages). Sage Publications, Inc.
- Kosina, L. (2017, May 3). Post, What percentage of new products fail and why? <https://www.publicity.com/marketsmart-newsletters/percentage-new-products-fail/>
- Kunz, M., Birr, S., Raslan, M., Ma, L., & Januschowski, T. (2023). Deep learning based forecasting: A case study from the online fashion industry, Chapter 11 In *Forecasting with artificial intelligence: Theory and applications*. Plaggrave.
- Lee, H., Kim, S. G., Park, H. W., & Kang, P. (2014). Pre-launch new product demand forecasting using the Bass model: A statistical and machine learning-based approach. *Technological Forecasting and Social Change*, 86(2014), 49–64.
- Loureiro, A., Miguéis, V., & Da Silva, L. F. (2018). Exploring the use of deep neural networks for sales forecasting in fashion retail. *Decision Support System*, 114(2018), 81–93.

- Luellen, E. (2020, July). Post the top five machine learning methods to forecast demand for new products, in towards data science. <https://towardsdatascience.com/the-top-five-machine-learning-methods-to-forecast-demand-for-new-products-b881ac7e3f32>
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*. <https://doi.org/10.1016/j.ijforecast.2021.11.013>
- Makridakis S., Spiliotis S., & Assimakopoulos V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward, Computer Science, Medicine.
- Makridakis, S., Wheelwright, S. C., & McGee, V. E. (1983). Forecasting—Methods and Applications, 2nd Edition. Wiley.
- Mas-Machuca M., Sainz M., & Martinez-Costa C. (2014). A review of forecasting models for new products. *Intangible Capital*, 10(1).
- Meade, N., & Al. (2000). Modelling diffusion and replacement. *European Journal of Operational Research*, 125, 551–570.
- Meade, N. (1984, October/December). The use of growth curves in forecasting market development—a review and appraisal. *Journal of Forecasting*.
- Menon S., & Ranjan R. (2020). Post The evolution of forecasting techniques: Traditional versus machine learning methods. <https://www.genpact.com/insight/the-evolution-of-forecasting-techniques-traditional-versus-machine-learning-methods>
- Ord K., Fildes R., & Kourentzes N. (2017). Principles of Business Forecasting, 2nd Edition. Wessex Press Inc.
- Schaer, O., Kourentzes N., & Fildes R. (2022, May). Predictive competitive intelligence with pre-release online search traffic. *Production and Operations Management*, 31(4).
- Semenoglou A., Spiliotis E., Makridakis S., & Assimakopoulos V. (2021, July–September). Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, 37(3), 1072–1084.
- Semenoglou, A., Spiliotis, E., & Assimakopoulos, V., (2023). Neural network ensembles for univariate time series forecasting. *Forecasting with Artificial Intelligence—Theory and Applications*. Palgrave.
- Singh P. K., Gupta Y., Jha N., Rajan A. (2019): Fashion Retail: Forecasting Demand for New Items, arXiv preprint arXiv:1907.01960.
- Smirnov P. S., & Sudakov V. A. (2021). Forecasting new product demand using machine Learning 2021. *Journal of Physics: Conference Series* 1925 012033.
- Sokele, M. (2008). Growth models for the forecasting of new product market adoption, in Telektronikk, 3 / 4.08. <https://nastava.tvvz.hr/kirt/wp-content/uploads/sites/4/2016/11/Telektronikk-2008.pdf>
- Thomasset, S., & Fiordaliso, A. (2006). A hybrid sales forecasting system based on clustering and decision trees. *Decision Support Systems*, 42(2006), 408–421.

- Thomassey, S., & Happiette, M. (2007). A neural clustering and classification system for sales forecasting of new apparel items. *Applied Soft Computing*, 7(2007), 1177–1187.
- University of Minnesota. (2016). Exploring Business. Author removed at request of original publisher. University of Minnesota Libraries Publishing, Edition 2016. <https://open.lib.umn.edu/exploringbusiness/> and <https://open.lib.umn.edu/exploringbusiness/chapter/10-1-what-is-a-product/>
- Van Steenbergen R. M., & Mes M. R. K. (2020, December). Forecasting demand profiles of new products. *Decision Support Systems*, 139.
- Vanston, L. (2008). Practical tips for forecasting new technology adoption, Telektronikk 3/4.2008. <http://tfi.com/pubs/forecastingtips.pdf>
- Vanston, L., & Hodges, R. (2004). Technology forecasting for telecommunications. *Telektronikk*, 100(4), 32–42. [http://tfi.com/pubs/w/pdf/telektronikk\\_peer.pdf](http://tfi.com/pubs/w/pdf/telektronikk_peer.pdf)
- Vanston, L., Rodgers C., & Hodges, R. (1995). Advanced video services—Analysis and forecasts for terrestrial service providers. Technology Futures, Inc., p. 106.
- Wright, M. J., & Stern, P. (2015). Forecasting new product trial with analogous series. *Journal of Business Research*, 68(2015), 1732–1738.

PART III

---

## Global Forecasting Models



## CHAPTER 5

---

# Forecasting with Big Data Using Global Forecasting Models

*Kasun Bandara*

## BIG DATA IN TIME SERIES FORECASTING

With the advancements of sensors and data storage capabilities, many businesses and industries nowadays routinely collect, manage and analyse vast amounts of time series data for better decision-making and strategic planning. This phenomenon is also known as “Big Data”, where companies have accumulated multitude of data over the years, which can assist them in landscaping their short-term, medium-term and long-term goals. Big data in a time series context does not necessarily mean that an isolated series contain more observations. This is because the sampling rate is application dependent and the length of a time series is dependent on the nature of the application, e.g., the sampling rate of half-hourly smart metre data is different to that of daily sales data. From the time series perspective, Big Data means that databases contain large collections of

---

K. Bandara (✉)

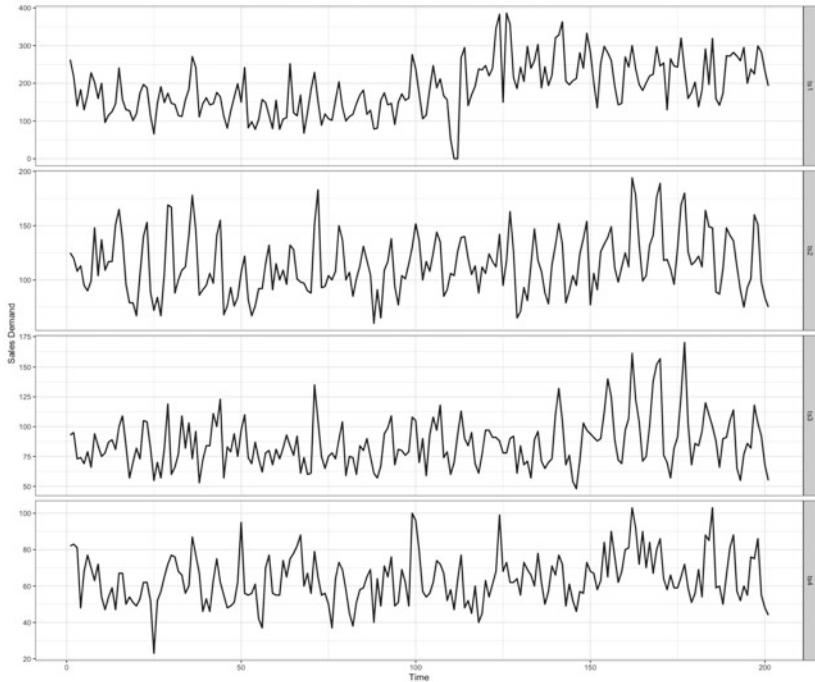
School of Computing and Information Systems, University of Melbourne,  
Melbourne, VIC, Australia  
e-mail: [kasun.bandara@unimelb.edu.au](mailto:kasun.bandara@unimelb.edu.au)

related time series that share key patterns in common. Examples of these are sales demand of related product assortments in retail, server performance measures in computer centres, household smart metre data and passenger demand of ride-share services. An example of such related time series is illustrated in Fig. 5.1, where sales demand of four different products are extracted from the M5 forecasting competition dataset. These four time series are selected from the bottom level of the sales hierarchy of the M5 dataset (Makridakis et al., 2022; M5 forecasting, 2021), where originally there exist sales demand time series of 30490 products. Therefore, modern enterprise databases are often consisted of large collections of related time series that may share key time series patterns and properties in common. In most cases, such properties can explain the latent features available in a certain domain, which can be useful to generate more accurate and meaningful forecasts.

Nonetheless, traditional (statistical) forecasting techniques such as exponential smoothing methods (Hyndman et al., 2008), auto-regressive moving-average models (Box et al., 2015) and prophet (Taylor & Letham, 2017) are univariate models that consider each time series separately, and forecast them in isolation. Consequently, these models are unable to exploit the cross-series information available in a group of related time series. Due to this reason, in the era of Big Data, forecasting approaches that build across sets of many time series are gaining popularity among the forecasting practitioners. In the time series forecasting literature, such models are referred to as Global Forecasting Models, which we discuss in the next section.

## GLOBAL FORECASTING MODELS

In contrast to univariate models, Global Forecasting Models (GFM) are simultaneously trained across sets of many time series. In GFMs, a single forecasting model is built using a group of time series; hence, GFMs are capable of borrowing the common patterns available in a collection of time series. Moreover, the model parameters of GFMs are estimated jointly using all the available time series (Januschowski et al., 2020).



**Fig. 5.1** Sales demand of four different products over a six months period, extracted from the M5 forecasting competition dataset, which is based on the daily product demand data from *Walmart*

The primary objective of global models is to develop an unified model  $F$ , which uses previous observations of all the available time series, i.e.,  $X = \{X_1, X_2, \dots, X_p\}$ . Here  $p$  refers to the number of time series in the dataset. The global model  $F$  can be formally defined as follows:

$$X_i^M = F(X, \theta) \quad (5.1)$$

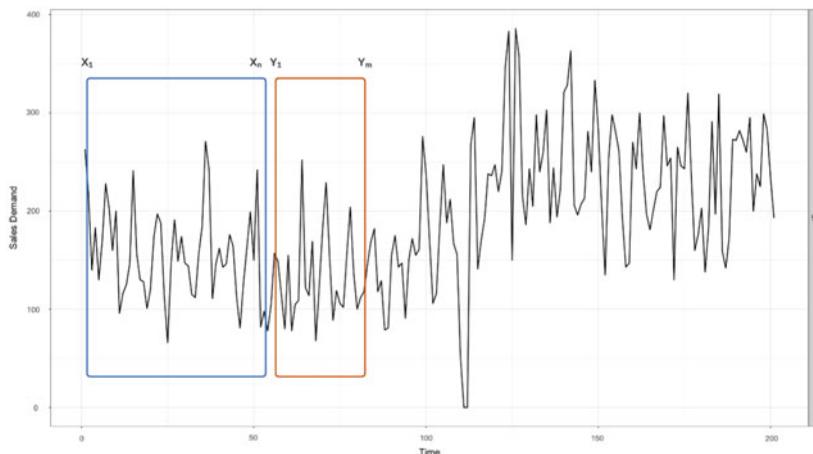
Here,  $M$  refers to the intended forecast horizon, whereas  $i$  is the target time series to forecast. As seen in Eq. (5.1), the model parameters ( $\theta$ ) of  $F$  is determined by using the information across all the available time series ( $X$ ). For example, if a neural network is used as the base model in  $F$ ,  $\theta$  represents the model weights, bias, whereas if a linear model is used,  $\theta$  represents the model coefficients. Therefore, depending on the

base model used in  $F$ , the values relevant to  $\theta$  changes. On the other hand, in a univariate forecasting solution,  $p$  individual functions, i.e.,  $F = \{F_1, F_2, \dots, F_p\}$  is built for each time series separately. Therefore, the model parameters are estimated for each time series separately.

### *Data Preparation for Global Models*

To build a global model, firstly, the past observations of time series are transformed into multiple pairs of input and output frames. This procedure is known as the moving window approach in the literature (Bandara et al., 2020; Hewamalage et al. 2021), where the generated input and output batches are later used to train and build global models.

Figure 5.2 shows an illustration of applying the moving window approach to a daily sales time series from the competition dataset. In this example,  $\{x_1, x_2, \dots, x_n\}$  represents the input window, whereas  $\{y_1, y_2, \dots, y_m\}$  corresponds to the output window. Here,  $n$  refers to the size of the input window, and  $m$  is the size of the output window. In a direct forecasting setting, where the predictions for the entire forecast horizon are produced at once, the size of  $m$  is set to the length of the intended forecast horizon, i.e.,  $m$  becomes equivalent to



**Fig. 5.2** An example of applying the moving window approach to a time series from the M5 forecasting competition dataset

$M$ . This approach is also referred to as the Multi-Input Multi-Output (MIMO) strategy in the forecasting literature (Ben et al., 2012). Whereas in a recursive forecasting setting (Ben et al., 2012), which uses the prediction from the last time step as input for the next prediction, the  $m$  is set to 1, where a sequence of one-step-ahead forecasts are generated to obtain the required forecast horizon  $M$ .

Figure 5.3 shows the output of the generated input-output window matrix after applying the moving window approach to a single time series length of  $k$ . This process is repeated to a group of time series and the resultant input and output window tuples are then pooled together to build a global model.

Both linear and non-linear models are suitable candidates to fit to the generated training data. In section “A Brief History of Global Models”, we provide examples of linear and non-linear models that have been used to build global models in the literature. Moreover, global models are particularly useful in situations where individual time series are too short to model effectively, i.e., the amount of information that can be extracted from a single series is limited. This is because global models are trained across sets of many time series using the input and output tuples generated from multiple time series. So even if the individual time series are short, pooling information across multiple time series can make more data available for model fitting.

Global models can be applied to a group of time series, even though the lengths of those time series are different from each other. If the time series lengths are different, the amounts of input and output windows generated for each time series are different. However, this does not preclude the training procedure of global models. This is a major difference between GFMs and multivariate forecasting models, where global models are not required to have all time series to align with each other in terms of the start and the end dates. Here, multivariate forecasting models refers to methods that assume there exist a interdependence between a set of time series, where the value of one time series is driven by multiple time varying variables. In multivariate forecasting models, such relationships are directly modelled and the predictions are produced for multiple time series variables simultaneously. Such interdependence between time series is not accounted in global models, where they independently generate forecasts on individual series, in which the model parameters are estimated using all the time series available (Januschowski et al., 2020).

$x_1$	$x_2$	$x_3$	$x_4$	...	$x_n$	$x_{n+1}$
$x_2$	$x_3$	$x_4$	$x_5$	...	$x_{n+1}$	$x_{n+2}$
				...		
				...		
$x_{k-n}$	$x_{k-n+1}$	$x_{k-n+2}$	$x_{k-n+3}$	...	$x_{k-1}$	$x_k$

**Fig. 5.3** The input-output window matrix after applying moving window approach to a time series length of  $k$ . Here, the size of the input window is  $n$  and the output window size is 1

## A BRIEF HISTORY OF GLOBAL MODELS

The competitive performance in recent forecasting competitions, such as CIF2016 Forecasting challenge (Štěpnička & Burda, 2017), Wikipedia Web Traffic Forecasting Challenge (Web Traffic Time Series Forecasting, 2018), M4 Competition (Makridakis et al., 2018), M5 Competition (Makridakis et al., 2022) have shown that GFM can be a competitive alternative to the traditional univariate statistical forecasting methods.

The use of global models has initially been introduced by Duncan et al. (2001), where those authors used a Bayesian pooling approach to extract information from a collection of time series. In this approach, authors use two models, a local model, which is estimated using the target time series that requires forecasts, and a global model, which is estimated using a pre-selected set of similar time series to the target series. The estimated local and group model parameters are then combined using Bayesian shrinkage formulas. The resultant parameters are used to estimate the final forecasts. A linear version of global models, also known as Pooled Regression models in the literature, developed by Trapero et al. (2015) was able to forecast sales demand of products with lesser historical data. This pooled regression model can be interpreted as a linear auto-regressive GFM that shares all parameters across many series. In year 2016, a

non-linear version of global models proposed by Štěpnička and Burda (2019) won the CIF2016 Forecasting challenge, outperforming many state-of-the-art univariate models, highlighting their promising performance. The winning solution of CIF2016 is a globally trained Recurrent Neural Network (RNN), which is a special type of neural network suited for modelling sequence related tasks, such as time series forecasting. A globally trained RNN based encoder-decoder architecture was implemented by Suilin (2018) in year 2018 to win the Wikipedia web traffic forecasting challenge organised by Google. This competition involved accurately forecasting the future web traffic of 145,000 Wikipedia articles.

Consequently, many forecasting researchers and practitioners started to investigate the use of global models for forecasting. In year 2018 Salinas et al. (2020), proposed a global auto-regressive RNN based framework named DeepAR, which is also capable of producing probabilistic forecasts. In the same year, the results of the renowned M4 forecasting competition showcased the superiority of global models in the presence of large collection of time series. The M4 forecasting competition was based on 100,000 real-world time series sourced from multiple application domains with various data frequencies. The winning submission of the M4 competition, Exponential Smoothing-Recurrent Neural Network (ES-RNN) (Smyl, 2019), uses a globally trained RNN with an ensemble of specialists methodology to forecast the competition time series. In year 2020, the M5 forecasting competition Makridakis et al. (2022), which is the next iteration of M competitions, was also won by a GFM-based solution, which used a gradient-boosted tree as the main predictor in the global model. In the M5 competition, it was required to accurately forecast 42,840 hierarchically organised time series, representing the sales demand of 3049 products sold by Walmart, a retailer in the United States. In addition to the winning solution, many other GFM-based submissions were able to secure top ranks in the M5 competition. These results highlighted the variety of base models that can be employed as the predictor model, i.e., RNN, gradient-boosted trees, with global models.

In recent years, more advanced forecasting architectures, which are mainly based on machine learning frameworks, have been introduced for global models. Oreshkin et al. (2020) proposed NBEATS framework that trains deep neural networks globally to achieve state-of-the-art accuracy on the M4 competition dataset. Also, Rangapuram et al. (2018) introduced deep state-space models that parameterise a linear state-space model with globally trained RNNs. Many neural network architectures

that were originally proposed for natural language processing (Transformer architecture Lim et al., 2021) and audio processing applications (Wavenet architecture Sen et al., 2019) have been recently adapted for time series forecasting in the form of global models. As these neural network architectures are often complex and contain numerous model parameters to optimise, it is necessary to have substantial amounts of training data to avoid possible model over-fitting. GFMs are beneficial to serve in such purposes, as they are trained across many time series together, providing adequate amounts of observations to fine-tune the model parameters.

## MODEL COMPLEXITY IN GLOBAL MODELS

The empirical insights from recent studies and forecasting competitions showcase the competitive nature of global models in the presence of large amounts of time series. As global models exploit more data across multiple time series, they can afford to be more complex, without being prone to overfitting. In other words, when more data is available, model parameters of advance prediction algorithms, i.e., machine learning based techniques, can be estimated more reliably compared to univariate models. As such, more complex forecasting algorithms can be employed as the base model in global models. Another favourable characteristic of global models is that the overall complexity of global model stays the same, when dataset grows. On the other hand, the complexity of univariate model increases as the dataset get bigger. For instance, consider fitting a simple univariate model that contains 5 model parameters, to a single time series. If the time series database contains 10,000 time series, the amount of parameters to be tuned will total up to 50,000, i.e.,  $5 * 10,000$ . Alternatively, if a complex machine learning based forecasting algorithm that contains 5000 model parameters, i.e., relatively complex than the univariate model, employ as a global model, the total parameters to be tuned will still equivalent to 5000, as a single model is trained across all the available time series. So, even though the model complexity of a single model is high, when employed as a global model, the total model complexity remains the same, which is independent of the number of time series.

More theoretical insights about the model complexity of global models have recently been discussed by Montero-Manso and Hyndman (2021), where those authors highlight that for larger datasets, global models can

be designed with a higher complexity, yet still achieve better generalisation error compared to univariate models. Furthermore, they argue that the complexity of global models can be increased by three different ways: (1) using long model memory models, (2) applying data partitioning and (3) employing non-linear, non-parametric models as the base model of global models.

**Long Model Memory** This can be achieved by adding more lags as inputs to the model. For example, if the base model of global model is an auto-regressive model, i.e., similar to the pooled regression model discussed in section “[A Brief History of Global Models](#)”, a higher complexity can be reached by increasing the memory/order of the auto-regressive model. In addition to better model fitting, this also enables the global model to capture long memory patterns, such as trends that are useful to generate accurate forecasts. As an empirical evidence, one of the top-ranked solutions of the M5 forecasting competition (Bandara et al., [2021](#)) uses an order of 400 sales lags, i.e., sales history of past 400 days, to produce 28 days ahead sales forecasts.

**Data Partitioning** Prior to fitting a global model across all the available time series, the time series database can be partitioned into multiple sub-groups, based on their similarity. So, instead of fitting a single global model, separate global models are fitted to each subgroup of time series. This procedure can be particularly useful if the time series database is heterogeneous, where prior grouping process is helpful to build a notion of similarity between the time series into global models. For example, Bandara et al. ([2020](#)) proposed a clustering based framework to group time series. Here, those authors used a feature-based clustering technique, where a fixed set of time series features are extracted from each series first and then traditional clustering algorithms, i.e., K-Means, DB-Scan, are applied to the extracted features. Based on the resultant grouping, a different global model is fitted to each cluster of time series. Alternatively, the grouping can also be done based on the additional domain knowledge available about time series. Several studies used the natural grouping available in a sales product hierarchy, i.e., based on product department/category/sub-category, to group time series, before fitting the global models (Bandara et al., [2019, 2021](#)). These studies

have empirically shown that prior time series grouping can be helpful to improve the accuracy of global models.

**Non-linear, Non-parametric models** More complex prediction models can be introduced as the base model in global models. For example, machine learning and deep learning based forecasting algorithms, when used as the base model in global models, have shown competitive results in recent studies and forecasting competitions (Bandara et al., 2019; Makridakis et al., 2022; Salinas et al., 2020).

## GLOBAL FORECASTING MODEL FRAMEWORKS

With the increasing use of global models among forecasting practitioners, several open-source based forecasting frameworks have added global models as a model class to their application programming interface (API).

GluonTS package,<sup>1</sup> based on Python programming language, is the most popular framework available to evaluate multiple global model architectures (Alexandrov et al., 2020). This library provides off-the-shelf implementations of general global model architectures, i.e., implementations of non-linear models such as RNN, and more specialised global model architectures developed for forecasting, i.e., DeepAR, NBEATS, Wavenet and Transformers, etc. This framework has been facilitating researchers and forecasting practitioners to add global models as an additional benchmark for their model evaluation purposes.

Darts<sup>2</sup> is another Python library that offers many implementations of global models (Herzen et al., 2022). Similar to GluonTS, Darts has basic implementations and more specialised global model forecasting architectures, including the Temporal Fusion Transformer model. In addition to global models, Darts package also provides multiple implementations of classic univariate models such as ARIMA and ETS models. Darts follows a similar API to famous scikit-learn python package, where users can likewise use *fit()* and *predict()* functions to fit and generate predictions from a model.

<sup>1</sup> <https://ts.gluon.ai/stable>.

<sup>2</sup> <https://unit8co.github.io/darts/>.

## RECENT ADVANCES IN GLOBAL MODELS

The strong empirical evidence, supported by recent theoretical insights, have made global model an increasingly popular alternative to traditional univariate models. In the following, we discuss several developments of global models, which have recently become successful and shown competitive results.

- **Handling Data sparsity:** As highlighted earlier, global models outshines univariate models in a presence of many time series. However, time series datasets may have only a limited number of time series that can hinder global models to reach their full potential. Recent studies have introduced different methodologies to generate accurate forecasts with global models under these circumstances. Bandara et al. (2021) developed a data augmentation framework to artificially generate time series and transfer information from the newly generated time series to global models. This enabled global models to use sufficient amount of information for training and still outperform univariate models in less data-abundant settings.
- **Causal Inference:** Casual inference analysis attempts to capture the causal relationships of interventions and effects. This technique is often used to measure the effects of an intervention in an application. In this process, counterfactual prediction plays the role of predicting under the assumption of absence of intervention, for treated instances in the post-intervention period, which is then used to calculate the difference between the true observations. This is referred to as the causal effect, i.e., the effect of introducing an intervention. More recently, several studies have shown that the global models are a better class of models to generate counterfactual predictions, compared to univariate models, as it allows to model treated and control time series simultaneously over the pre-treatment period (Grecov et al., 2021, 2022). Moreover, compared to traditional approaches, global models obviates the need of making prior equivalence assumptions between distributions of the control and treated time series in the pre-treatment period, as both control and treated time series can be trained together in a unified setting.
- **Model Ensembling:** According to the recent forecasting literature, GFM<sub>s</sub> have also successfully employed in forecast ensembles along with the univariate models (Godahewa et al., 2022). Here,

the motivation is to capture the local behaviours of a time series using univariate or local models, while capturing the common patterns across time series using global models. This premise has recently been used in an energy forecasting competition (Triguero, 2020) to forecast household energy consumption. Here, one of the top-ranked submissions proposed an approach that use statistical univariate techniques to capture the unique energy consumption behaviours of households and global models to capture the common energy consumption patterns of households. Later, these forecasts are combined to improve the overall accuracy of the forecasts.

## CONCLUSIONS AND FUTURE WORK

As the use of global models are gaining interest among forecast practitioners and researchers, the challenges and innovation around the application of global models are also growing at a fast pace. Therefore, the opportunities in these models to improve the forecast accuracy and use for various real-world forecasting applications are far-reaching. Based on the recent studies, following can be identified as several research segments that require future work in global models.

- **Model Interpretability:** One of the advantages of using univariate, statistical models such as ETS, ARIMA is because of their model interpretability on the generated forecasts. In other words, their model forecasts can be explained based on the types of components used for model fitting, i.e., seasonality-type, trend type, lags, etc. However, the forecasts generated from global models are often difficult to interpret and explain, especially towards specific time series. To address this challenge, more recently, Rajapaksha et al. (2022) proposed a local model-agnostic interpretable forecasting framework to explain the forecasts generated from global models.
- **Hierarchical Forecasting:** Forecasting time series that are hierarchically organised with several layers of granularity is a common challenge faced by many businesses and industries. The traditional approach to forecast such time series is to first generate individual forecasts for each time series (mostly using a univariate technique) in the hierarchy, and then apply a reconciliation algorithm to ensure the coherency of the forecasts across the hierarchy. On the other

hand, as global models can be trained across the entire hierarchy and better position to exploit cross-series information in a hierarchy compared to traditional univariate methods. Furthermore, as discussed in section “[Model Complexity in Global Models](#)”, global models are scalable to increasing volumes of time series data, and can better handle large-scale hierarchical time series databases. More recently, several studies have attempted to use a unified approach to forecast hierarchical time series (Han et al., [2021](#); Mishchenko et al., [2019](#)).

- **Tree based regression models:** As discussed in section “[A Brief History of Global Models](#)”, tree based regression models such as gradient boosting models, are becoming increasingly competitive for forecasting, when trained as global models. However, these tree-based models are general-purpose models that do not primarily focus on forecasting. Therefore, more forecasting-specific tree based algorithms can be introduced to produce better forecasts in this space (Godahewa et al., [2023](#)).

## REFERENCES

- Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., & Wang, Y. (2020). GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, 21(116), 1–6.
- Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140, 112896.
- Bandara, K., Hewamalage, H., Godahewa, R., & Gamakumara, P. (2021, December). A fast and scalable ensemble of global models with long memory and data partitioning for the M5 forecasting competition. *International Journal of Forecasting*.
- Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y., & Bergmeir, C. (2021, December). Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognition*, 120, 108148.
- Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., & Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *Neural information processing* (pp. 462–474). Springer International Publishing.

- Ben Taieb, S., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012, June). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, 39(8), 7067–7083.
- Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. Wiley
- Duncan, G. T., Gorr, W. L., & Szczypula, J. (2001). Forecasting analogous time series. In J. S. Armstrong (Ed.), *Principles of forecasting: A handbook for researchers and practitioners* (pp. 195–213). Springer.
- Godahewa, R., Bergmeir, C., Webb, G. I., & Montero-Manso, P. (2022, March). An accurate and fully-automated ensemble model for weekly time series forecasting. *International Journal of Forecasting*.
- Godahewa, R., Webb, G. I., Schmidt, D., & Bergmeir, C. (2023). Setar-tree: A novel and accurate tree algorithm for global time series forecasting. *Machine Learning*, forthcoming.
- Grecov, P., Bandara, K., Bergmeir, C., Ackermann, K., Campbell, S., Scott, D., & Lubman, D. (2021). Causal inference using global forecasting models for counterfactual prediction. In *Advances in knowledge discovery and data mining* (pp. 282–294). Springer International Publishing.
- Grecov, P., Prasanna, A. N., Ackermann, K., Campbell, S., Scott, D., Lubman, D. I., & Bergmeir, C. (2022). Probabilistic causal effect estimation with global neural network forecasting models. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.
- Han, X., Dasgupta, S., & Ghosh, J. (2021). Simultaneously reconciled quantile forecasting of hierarchically related time series. In *AISTATS*.
- Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Pottelbergh, T. V., Pasieka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kołacisz, J., Bader, D., Gusset, F., Benhedi, M., Williamson, C., Kosinski, M., Petrik, M., & Grosch, G. (2022). Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124), 1–6.
- Hewamalage, H., Bergmeir, C., & Bandara, K. (2021, January). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1), 388–427.
- Hyndman, R., Koehler, A. B., Keith Ord, J., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: The state space approach*. Springer Science & Business Media.
- Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1), 167–177.
- Lim, B., Arik, S. Ö., Loeff, N., & Pfister, T. (2021, June). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37, 1748.

- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34(4), 802–808.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022, January). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*.
- Mishchenko, K., Montgomery, M., & Vaggi, F. (2019). A self-supervised approach to hierarchical forecasting with applications to groupwise synthetic controls. In *ICML*.
- M5 forecasting—accuracy. <https://www.kaggle.com/competitions/m5-forecasting-accuracy/data>. Accessed 12 October 2022.
- Montero-Manso, P., & Hyndman, R. J. (2021, June). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*.
- Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *ICLR*.
- Rajapaksha, D., Bergmeir, C., & Hyndman, R. J. (2022, August). LoMEF: A framework to produce local explanations for global model time series forecasts. *International Journal of Forecasting*.
- Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc.
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020, July). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.
- Sen, R., Yu, H.-F., & Dhillon, I. S. (2019). Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc.
- Smyl, S. (2019, July). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*.
- Štěpnička M., & Burda, M. (2017). On the results and observations of the time series forecasting competition CIF 2016. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1–6), July 2017.
- Suilin, A.. (2018). Kaggle-web-traffic. <https://github.com/Arturus/kaggle-web-traffic>. Accessed 10 February 2020.
- Taylor, S. J., & Letham, B. (2017, September). Forecasting at scale. Technical Report e3190v2. PeerJ Preprints.

- Trapero, J. R., Kourentzes, N., & Fildes, R. (2015). On the identification of sales forecasting models in the presence of promotions. *Journal of the Operational Research Society*, 66(2), 299–307.
- Triguero, I. (2020). IEEE-CIS technical challenge on energy prediction from smart meter data.
- Web Traffic Time Series Forecasting. <https://www.kaggle.com/c/web-traffic-time-series-forecasting>. Accessed 12 October 2022.



## CHAPTER 6

---

# How to Leverage Data for Time Series Forecasting with Artificial Intelligence Models: Illustrations and Guidelines for Cross-Learning

*Pablo Montero-Manso*

## INTRODUCTION

Artificial Intelligence models are complex, highly flexible but also notoriously data hungry. AI models are trained on datasets of millions of observations in order to achieve their outstanding performance. Historically speaking, time series forecasting has resisted the advances of AI compared to other domains that have become dominated by it, such as Computer Vision, Speech Recognition or Natural Language Processing. The resistance of forecasting has to do with the low sample size and high stochasticity present in many forecasting problems. Suppose one wants to forecast the time series of GDP growth of a country. The information

---

P. Montero-Manso (✉)

The University of Sydney, Sydney, NSW, Australia

e-mail: [pablo.monteromanso@sydney.edu.au](mailto:pablo.monteromanso@sydney.edu.au)

in this time series is measurements of GDP per year, going back to the 1950s, so we would have about 70 observations for this time series by the 2020s. The traditional way of forecasting this time series, which we would call the local way, is to fit a model to the data in this time series. 70 observations are hardly enough data to train modern AI models. Moreover, unlike other fields, where more data can be acquired at a cost (e.g. sampling more images in Computer Vision), it is not possible to increase the data in this time series, and no data exists prior to 1950. Even if we were to measure the GDP with more frequency, such as once per quarter, we would have 280 observations, still a small amount of data. It is not even clear that this new data would be relevant (consider the extreme case of measuring the GDP every day; would this help improve predictions of GDP in future years?).

Due to the critical importance of time series forecasting, much research effort has been devoted to developing models specific for this task. The models that have come out of these efforts are based on simple yet powerful statistical assumptions about the underlying nature of the data. For example, the celebrated families of models Exponential Smoothing, ARIMA(Hyndman & Athanasopoulos, 2018) or Theta(Assimakopoulos & Nikolopoulos, 2000) models use very few parameters; they are quite restrictive in the patterns they can express, but this small set of patterns is actually very useful in real-world applications. Looking at benchmark datasets, these model families have dominated in predictive accuracy(Makridakis & Hibon, 2000) and are still top performers in many forecasting tasks (Rakshitha et al., 2023).

Despite the impressive performance of simple models, it is safe to assume that *at least some* time series should be more complex than what existing simple models impose. The complexity could be even beyond any possible simple model that could be designed by domain experts. As in many other fields that have come to be dominated by AI models (e.g. Image Recognition), AI models are capable of picking this complexity to outperform other methods. The obstacle remains in how to train AI models in this low sample setting.

This chapter analyzes methods to overcome the low sample problem that have been named as “Cross-learning” or “Global Models” or “Data pooling”.

Cross-learning is a paradigm for training time series forecasting models that works by fitting a model to “across” group of time series. All series

in the group are forecasted by the same predictive function, a function “global” to all time series in the set (see section “[Background and Definitions](#)” for a formal definition). The underlying assumption in cross-learning seems quite strong: all time series in the set follow the same pattern. Assuming that all time series follow the same pattern, then we can pool together the data from all time series and try to find this pattern (e.g. via an AI model) using the data in *all* time series, not by fitting one AI model per time series (what is called the “local” method).

Cross-learning contrasts with the traditional way of forecasting, where each time series is assumed to follow its own ‘local’ pattern that might be different from the others. Intuitively, because cross-learning assumes similarity in how series are predicted, it is well suited to forecasting sets of similar time series, but it is not clear what would happen when the sets of time series are highly heterogeneous.

Cross-learning outperforms traditional local approaches in an (increasing) number of scenarios. However, the practice of cross-learning lacks fundamental principles; as a result, the amount of trial and error involved is even greater than what is commonly expected from AI modeling.

This chapter aims at providing explanations on the mechanisms underlying the performance of cross-learning, delivering insights as well as guidelines on how to effectively apply the cross-learning approach with AI models to improve forecasting accuracy. The exposition proceeds at two levels, general results and specific results.

The following two general results deal with the application cross-learning to large datasets, and from these results, some guidelines can be derived. The results are:

- Cross-learning and local learning can produce the same forecasts; cross-learning is not restrictive. This result makes cross-learning widely applicable; it can work even in huge datasets of widely different series. The technical aspects of the equivalence, however, hint that cross-learning model classes should be more complex than the typical local models, both in the number of lags (how many past observations to use) and in feature transformations, leading to the use AI techniques in this setting. This additional complexity is needed even when the underlying processes are simple, therefore motivating the use of AI even for datasets that might seem suitable for traditional models. Section “[Approximation Capacity](#)

of Cross-Learned Models” discusses the equivalence and its consequences.

- Statistical trade-offs that compare the forecast error as a function of model complexity (i.e. number of parameters or flexibility of the model) and size of the set of time series. Cross-learning can improve the performance the more the time series are considered simultaneously, while local does not. The trade-offs point to the possibility of success for extremely large AI models. The result also indicates that the dataset itself is an important component that should be tuned: Practitioners can manipulate the set of time series to alter performance. Trade-offs and recommendations derived from them are covered in section “[Statistical Trade-Offs](#)”.

For the specific results, we analyze examples of the most typical idealized time series processes that can appear in practice, such as logistic growths, periodic patterns or linear trends. We then derive explicit formulas for cross-learned models that give exactly equivalent results for these processes. The examples are specific to a limited set of processes, but they also illustrate some principles that could be applied in other contexts. The mathematical tools for this analysis are linear functions, which might seem at odds with Artificial Intelligence methods that are highly nonlinear. The purpose is to give completely detailed examples that can be easily understood. The main motivation for AI comes from the fact that real processes are close to these idealized processes, but differ in ways that are difficult to define manually, so AI is perfectly suited to capture that extra complexity. Moreover, particular to forecasting, many real time series problems can actually be solved by linear or very simple models. The issue in time series is that finding the right linear model given data is a difficult endeavor, for which Artificial Intelligence is extremely helpful. The specific results covered are:

- Section “[Illustration of Cross-Learning Approximation of Specific Time Series Processes](#)” is devoted to the explicit analysis of how a set of many diverse time series processes can be modeled with a single function, and how this function looks like. The analysis covers typical processes such as exponential or logistic growths, periodic patterns or lineal trends. This provides a complete mathematical understanding of the mechanism underling the equivalence between local

and cross-learning models for these families of processes. The analysis is of idealized processes and the extension to realistic applications makes a strong motivation for Artificial Intelligence models.

- Section “[Explicit Example of Transfer Learning](#)” shows an example of transfer learning or how some time series in the group can help forecast others. This deals with situations that might frequently happen in practice, such as the effects of an intervention have already happened in some series, but not in others, so the intervention can be cross-learned and benefit. The example analyzed is of COVID-19 predictions, where some countries were ahead of others in the progress of the pandemic and introduced lock-downs earlier. The information in these advanced countries can be transferred to the rest of the countries automatically via cross-learning.

## BACKGROUND AND DEFINITIONS

We establish some concepts and definitions that will be used in the chapter. We divide the section into two parts. The first part deals with basic definitions and concepts that are common in time series forecasting, the “building blocks” for the content of the chapter. It serves the double purpose of introducing the concepts and resolving ambiguities for the reader that might already be familiar with them. If these concepts are new, we suggest consulting the references provided. The second part focuses on the concepts of locality and globality (or cross-learning).

### *Common Terminology and Building Blocks*

- **Time Series:** Is a vector of real numbers of fixed, finite length. For example, a time series  $X$  of  $T$  observations is denoted by:

$$X = (X_1, X_2, X_3, \dots, X_T)$$

The observations of a time series come from taking measurements of the same natural phenomenon at a regular frequency.

- **Set of time series:** A collection of time series in no particular order. The time series can differ in length, though for simplicity we consider the length of all time series in the set to be the same. This simplification is not restrictive; for example, we could pad time series (by adding zeroes to the series) to match the length of the

longest in the set. A set of time series  $\mathcal{X}$  containing  $N$  time series of length  $T$  is denoted by:

$$\mathcal{X} = \{X_{i,t} \mid i = 1 \dots N \text{ and } t = 1 \dots N\}$$

so that  $X_{2,7}$  would represent the 7th observation in the 2nd time series in the set  $\mathcal{X}$ .

- **Predictive function:** Is the function used to compute the “future” values of a time series. In our context of time series forecasting, it is a function that takes some observations of a **single** time series and outputs a real number, the computed forecasts. We do not consider predictive functions that can take on covariates or observations from several time series.

We use the term **autoregression** or **autoregressive** predictive functions, to indicate that these are functions of fixed input length, and they are functions that predict future values based on a fixed amount of “past” observations. The number of past observations (input size of the function) is called **lags** or the **order** of the autoregression. It can be also called the **memory** of the model, since it measures how many past observations are needed to predict the series.

For example, an autoregressive function  $p$  of “3 lags”, “lag 3” or “order 3” could be denoted by:

$$X_{t+1} = p(X_t, X_{t-1}, X_{t-2})$$

Therefore, this implies that for predicting the 5th observation of a time series:

$$X_5 = p(X_4, X_3, X_2)$$

For the 7th observation:

$$X_7 = p(X_6, X_5, X_4)$$

If we desire to predict  $X_3$  with  $p$ , some workaround is required because the autoregressive function cannot be computed, since there is no  $X_0$ .

A simple workaround for this can be padding, as mentioned earlier. An autoregressive function contrasts with other ways of

predicting time series, for example as functions of time, say the function  $f$

$$X_t = f(t)$$

so that  $X_5 = f(5)$ . However, when modeling a time series as a function of time, it is more common to consider continuous time, not discrete time steps, so they are more commonly denoted as  $X(t) = f(t)$  (we do not cover continuous time in this chapter).

Autoregressive functions, because they are of fixed input, can be thought to be more restrictive than some varying input length predictive functions popular in time series forecasting, such as recurrent neural networks. For our analysis, we can consider just autoregressive functions without loss of generality: in practical situations with time series of finite length, setting the input of the autoregressive function to be the whole length of the series makes autoregression equivalent to varying input length functions. Again, we can pad the time series so that there are no problem with the definitions.

We consider only **single-horizon** predictive functions or functions that predict the next time step (e.g.  $X_{T+1}$  as a function of the observed values  $(X_T, \dots, X_1)$ ). Our results are also valid for **multi-horizon** predictive functions (e.g.  $(X_{T+3}, X_{T+2}, X_{T+1})$  as a function of  $(X_T, \dots, X_1)$ ), just by recursively applying the predictive function:

$$X_{T+1} = p(X_T, X_{T-1}, X_{T-2}, \dots)$$

$$X_{T+2} = p[p(X_T, X_{T-1}, X_{T-2}, \dots), X_T, X_{T-1}, \dots]$$

$$X_{T+3} = p\left(\begin{array}{l} p[p(X_T, X_{T-1}, X_{T-2}, \dots), X_T, X_{T-1}, \dots], \\ p(X_T, X_{T-1}, X_{T-2}, \dots), X_T \end{array}\right)$$

or by using a different predictive function for each time step.

- **Model class:** Is a set of predictive functions. **Fitting** or **estimating** a model means finding the “best” predictive function within the set, for example the predictive functions that minimize some measure of forecast error when applied to the time series of interest. The term “model” is often used ambiguously to represent either a single

predictive function, a whole model class, the estimation method or a combination of any of those terms. An example of model class can be the AR(2) (Hyndman & Athanasopoulos, 2018) the set of all autoregressive functions of order 2 that are linear combinations of the last two observations of a given time series, restricting the coefficients of the combination to represent a stationary (Hyndman & Athanasopoulos, 2018) process. We highlight another potential source of confusion: the term “AR” comes from AutoRegressive, but it refers only to a specific family of autoregressive functions, only linear functions, and not even all possible linear functions, but those that would generate stationary processes.

### *Locality and Globality*

We now move onto the formal definitions of the main object of analysis in this chapter, the two methods of modeling from the perspective of forecasting a set of time series.

- **Local modeling:** Corresponds to the “traditional” way of forecasting, fitting a model to each time series in the set of series. We can get a different predictive function for each time series in the set. A model class that is fit by trying to find the best predictive function within the set to forecast a **single** given time series.

For the sake of clarity and avoiding ambiguities, we give an specific example. Suppose that our model class is the set of linear autoregressive functions of order 3. This means that for a generic time series, the value for the next time step is modeled as a linear combination of the “last three” or the “three more recent” observations:  $Y_{t+1} = \phi_1 Y_t + \phi_2 Y_{t-1} + \phi_3 Y_{t-2}$ . In the context of modeling a dataset, local modeling finds the best values for the parameters for each series  $X_i$  in the set  $\mathcal{X}$ :

$$X_{i,t+1} = \phi_{i,1} X_{i,t} + \phi_{i,2} X_{i,t-1} + \phi_{i,3} X_{i,t-2}$$

Notice how the parameters  $\phi$  vary between time series (the subindex  $i$  as in  $\phi_{i,1}$ ).

- **Cross-learning modeling:** Also known as global modeling or data pooling. When we fit a cross-learning model, we try to find the best

single predictive function within a model class for all time series in the set. This is, all time series in the set will be forecasted by the same predictive function. This predictive function gets passed each time series in the set separately. Using the same dataset of time series  $\mathcal{X}$ , global modeling will find the best values of the parameters across all time series:

$$X_{i,t+1} = \phi_1 X_{i,t} + \phi_2 X_{i,t-1} + \phi_3 X_{i,t-3}$$

Notice how in the global or cross-learning method, all series are predicted by same linear combination; there is not subindex  $i$  in the parameters. The separation between local and cross-learning is not definitive; there is the possibility of “hybrid” learning. For example, we could learn some of the coefficients of the predictive function via cross-learning and others locally. The following shows how the first coefficient of the linear combination is local to each time series ( $\phi_{i,1}$ ), but the other two parameters are the same across series ( $\phi_2, \phi_3$ ).

$$X_{i,t+1} = \phi_{i,1} X_{i,t} + \phi_2 X_{i,t-1} + \phi_3 X_{i,t-3}$$

Examples in this chapter will compare purely local and purely global methods.

## APPROXIMATION CAPACITY OF CROSS-LEARNED MODELS

This section will show that cross-learning can succeed in forecasting arbitrary datasets of time series, for both in terms of the number of time series in the set and in how different they can be. Local methods train many predictive functions, one per series, and global methods train a single predictive function for the whole dataset, but both methodologies can produce the same forecasts. It is a result that might seem counter-intuitive because cross-learning applies a single function to all time series, so it is difficult to envision how the same function can work well for a large number of heterogeneous time series. The results states:

Local and cross-learning approaches can produce the same forecasts for any given set of time series. Cross-learning models might require higher length of autoregression (more lags), or a larger model class. (more variable transformations)

What follows is an informal explanation of the reasoning behind the result; details and formal statement of it can be found in Montero-Manso and Hyndman (2021).

In cross-learning, we fit a model to a set of time series, obtaining a single forecasting function that will be applied to all time series in the set. This contrasts with the traditional local approach that fits a model to each series and potentially uses a different predictive function per series. At some level, we can consider that cross-learning is more restrictive than local learning, simply because cross-learning is forced to use a single function, and local learning can use as many predictive functions as series in the set. Consider the example: We have a set of two time series,  $A$  and  $B$ . Suppose each is generated from a “true” data generating process (DGP) for the linear autoregressive of order 2 family, but with different coefficients. For example, observations for these processes are generated following the equations:

$$A_{t+1} = 0.3A_t - 0.1A_{t-1} + \varepsilon_t$$

$$B_{t+1} = 0.2B_t + 0.15B_{t-1} + \eta_t$$

where  $\varepsilon_t$  and  $\eta_t$  are noise processes (each time step  $t$  is an independent sample coming from a random distribution).

A local model that fits linear autoregressive models of order 2 could theoretically recover the true coefficients for each process if given enough data. A global approach that fits a linear autoregressive models of order 2, on the other hand, is limited because it needs to express two different processes using the same values for the coefficients. This is, the best possible values for the coefficients for predicting  $A$  are  $(0.3, -0.1)$ , and the best possible coefficients for forecasting  $B$  are  $(0.2, 0.15)$ . Cross-learning has to use one set of coefficients so it will never be able to predict as accurate as local. Basically, a single predictive function can never reproduce two different processes.

This example might lead us to believe that cross-learning could be effective only in some situations, such as when the true processes behind  $A$  and  $B$  have very similar coefficients. But there is no good cross-learning solution for the previous example, so we could suppose that cross-learning is less generally applicable than local learning. This limitation could be even more relevant in the case of increasingly large datasets, because we expect them to have more different time series and we still

can only use one predictive function. The more time series in the set, the less likely that they will follow similar processes, so the less likely that cross-learning would be effective.

However, the limitations of cross-learning compared to local learning disappear when we consider two arguments:

1. **The model class for cross-learning can be larger than the model class used for local.** This means, for example, using higher-order autoregression (predictive functions that use more past observations, more lags) for the global model class. It also means that the model class used for cross-learning includes the local one; for example, the local could be linear autoregressive, and the global could add non-linearities (e.g. squared transformation, or a neural network). However, this condition is **not sufficient** to guarantee that a cross-learning model can succeed. The local model class (even the true one) could be already very complex, using a large set of functions with all possible lags. So just using a highly complex model class for cross-learning, such as larger deep neural network, does not guarantee that cross-learning will succeed.
2. **The predictive functions for both local and global have to be *estimated* from data.** This is the practical scenario: local and global are not ‘oracle’ methods where the best predictive functions are just given. Both global and local models have to be estimated (trained), and the procedure of estimation itself is also a *function of the data*. Estimation applies the same function to each time series when local modeling. For example, when we estimate by minimizing a loss function, say by the algorithm of least squares estimation, we apply the same algorithm to all time series in the set. From this point of view, both global and local use a single function for predicting, in fact, local modeling is a special case of global modeling.

### Guidelines relative to model capacity

This equivalence between local and global in terms of forecasting capacity is a simple result, but it can be used to derive good cross-learning model classes and practices, or to analyze existing practices.

- Cross-learning models have the same applicability as local models, and cross-learning is not “restrictive”. As a consequence,

there are no requirements to be made on the nature of the set of time series, neither its size nor the “similarity” between series in the set. However, this statement does not guarantee equivalent empirical performance in all situations, just potential of application. Without this result, for example, we would not be applying cross-learning to arbitrary sets of series, only to datasets could subjectively considered “related”.

- **Model classes for cross-learning should be more complex than local if we want to guarantee equivalent capacity.** Cross-learning model classes should essentially be a super set of traditional local model classes. They should include predictive functions with more lags or more nonlinearities or variable transformations. This motivates the use of highly complex or nonlinear AI model classes such as neural networks or decision trees for cross-learning on large datasets.
- **The increase in complexity required in cross-learning has consequences for finding good model classes.** There is no clear way on how to expand traditional model classes that already work well in the local approach, such as Exponential Smoothing, to cross-learning. Therefore, it is difficult to express “known properties” of the time series processes underlying the data in cross-learning model classes. For example, we might know that the processes we want to predict, or some of them, are “seasonal” (express periodic patterns, such as yearly cycles). Dealing with cyclical patterns is a solved problem with local models, but the cross-learning counterpart is not obvious; it is not easy to design a model class that will “pick” those patterns in a heterogeneous dataset when we apply cross-learning. Part of this chapter is devoted to this problem; we will show how a range of time series models derived from the fundamental nature of the processes (e.g. physics, economics, epidemiology) can be adapted to cross-learning. We will give an analysis for some specific model classes in section “[Illustration of Cross-Learning Approximation of Specific Time Series Processes](#)”.
- **The increase in complexity required in cross-learning has consequences for interpretability.** Even if cross-learning can deliver superior forecast accuracy, predictive functions coming out of it will be more difficult to interpret. Cross-learned functions will have more parameters or will provide less insights compared to local

models. Forecasting is often embedded within a larger decision-making process for which interpretation of the models that provide the forecast is key. However, there are advances in interpretability of global models via fitting an interpretable local model that mimics the cross-learned process (Rajapaksha et al., 2022).

- **There is no need for identifying series or group of series in the set to guarantee the approximation equivalence.** Our result shows equivalence between local and cross-learning; however, there is no actual need for this result if we apply some ‘workarounds’ or transformations to the dataset. A very popular workaround (Salinas et al., 2020) is to add synthetic variables that identify each series in the set. For example, we can add a co-variable to each time series that makes it unique, such as assigning an ID number to each series and then adding the dummy encoding of the ID as co-variate to each series. However, these workarounds can make the cross-learning model *too* equivalent to a local model. Consider the linear autoregressive model class, adding dummy binary variables to identify each time series in a global model. If these dummy variables are added via interaction terms with the main autoregressive coefficients, we get *exactly* a linear local autoregressive model (recall classic linear regression, interaction terms essentially make the model fit several linear models, one per group indicated in the dummy encoding). Therefore, in this case, the potential for better cross-learning would be lost by “overidentification”. From this analysis comes a simple pragmatic recommendation: Do not to start by identifying each series to make cross-learning have the same capacity as local, since it can make cross-learning behave exactly as local, removing the benefits that cross-learning brings to the table. Identification might work in cross-learning, but we recommend starting our modeling process by not identifying, then if this does not work, try identifying whole subsets of series according to some property (some metadata of the series), and then identifying the each individual time series as the last resort.

## STATISTICAL TRADE-OFFS

We have seen that cross-learning can approximate as well as local. This justifies its general application and allows us to derive some recommendations about how it should be applied, but gives us no information

in terms of empirical forecasting performance. This is, we know that both local and global are *a priori* equally capable in all situations, but we do not know when we should apply one or the other, what performance should we expect given a specific dataset. Introducing statistical assumptions about the data can give rise to additional insights and can guide the modeling process with cross-learning. In this section, we give a simple result that compares cross-learning modeling approaches to local in terms of statistical forecast performance. The result relates five main components:

1. **Model class complexity:** The complexity of a model class increases with the size of the set of functions. The more patterns a model class can express, the more complex it is. We denote complexity by  $C$ . For the sake of comparison, we assume in our analysis that local and cross-learning use the same model class. Model class complexity is applicable to the majority of models used in data science, including the popular AI models such as deep networks.
2. **Training forecast error:** The training error resulting from estimating the best predictive function within the model class, averaged across all time series in the set. Cross-learning and local training errors are denoted  $E_{\text{train}}^{\text{Cross}}$  and  $E_{\text{train}}^{\text{Local}}$  respectively.
3. **Size of the set of time series:** Denoted by  $N$ . Assume the time series are independent.
4. **Local information:** A quantity that can be considered proportional to the length of the time series. The longer we observe a time series, the more information we have about it and the better the predictions would be. We denote local information by the term  $T$  (the same as the length of the series, for simplicity) and is assumed to be the same for all series in the set.
5. **Average forecast performance:** The expected forecast error on the test set, averaged across all time series in the set. This is the main quantity of interest; all other components combine to give us an approximation on how the model is going to perform. Cross-learning and local performances are denoted  $E_{\text{test}}^{\text{Cross}}$  and  $E_{\text{test}}^{\text{Local}}$  respectively.

With these terms, we get two expressions for the performance based on the other components, one for cross-learning and another for local,

that we can analyze for insights (the expressions are a simplification of a result in Montero-Manso and Hyndman (2021)).

$$E_{\text{test}}^{\text{Cross}} \leq E_{\text{train}}^{\text{Cross}} + \mathcal{O}\left(\sqrt{\frac{C}{NT}}\right)$$

$$E_{\text{test}}^{\text{Local}} \leq E_{\text{train}}^{\text{Local}} + \mathcal{O}\left(\sqrt{\frac{C}{T}}\right)$$

The result can be interpreted as follows: the test error increases with the training error and model complexity, while it decreases with the size of the set and the local information (length of the time series). The result can be understood in a similar way to the Bias-Variance Tradeoff in Statistical Learning: large model complexity reduces the bias, but increases the variance. In our case, model complexity reduces the training error but increases test error and hence these two quantities trade off.

The main highlight when comparing cross-learning and local is that local models do not improve with the size of the set. On the other hand, for the same model class, local approach will usually have much smaller training error so this result does not imply that cross-learning dominates a local approach, only how performance relates to the size of the set of time series for a given training error. When local information  $T$  is relatively high, both approaches will get similar performance.

We highlight several limitations about the strength or applicability of this result:

- The result refers to worst-case scenario performance, which can be quite far away from actual performance in any particular problem.
- The errors are averaged across time series in the set; this result does not specify individual errors.
- To arrive at this result, we need to make some assumptions such as time series in the set being independent, known model complexities and local information, and others. The assumptions are violated in almost all realistic situations, time series are not independent, models are fit by advanced estimation methods (regularization, stochastic optimization) so actual model complexity is difficult to calculate, time series has varying degrees of local information and training errors cannot be known beforehand. Therefore, the result

should be used as a guide for modeling, not as a strict quantitative relationship that will hold in practice. Moreover, modern AI/Machine Learning tends to apply large model classes that can fit the training data perfectly. In these situations, the bounds become non-informative (e.g. expected average test error indicated by our result is larger than the maximum possible in the domain).

### Guidelines derived from statistical trade-offs

In the result, we identify the “ingredients” or components that affect the statistical performance. As modelers and practitioners, some of these components are under our control. We will analyze the effect of the components model class and the set of time series. Adjusting the model is a very common practice across all data science disciplines. Adjusting of datasets is less common; for example, local models do not benefit from it.

- **Global model classes can afford to be much more complex than local model classes.** They can include more lags and nonlinearities and still generalize relatively well. Since the performance of cross-learning improves with the number of time series, a very large dataset will allow for very large model classes, which would be impractical for local approaches. This gives a rationale on the good empirical performance of complex AI models in cross-learning and not in local, while also motivates the use of “counter-intuitively large” model classes, such as trying models with much more lags than we apply to local.
- **Cross-learning has an additional *tuning parameter* compared to local learning: the number of series in the dataset.** The suitability of a model class to a given set of time series can be measured by how much the training error improves when increasing the complexity of the model class, how much error degrades when reducing the complexity of the model class. This is valid for both local and cross-learning and follows typical guidelines in data science. However, in cross-learning, **altering** the set of time series also affects training error. In other words, changing the dataset does not affect the performance of local models, but it affects the performance of global models. We can think that cross-learning has an additional “dimension” or tuning parameter; we can adjust the set of time series, for

example by applying cross-learning to separate subsets of the set of interest, or by adding external time series to the set. Specifically:

- **Separating datasets: Clustering.** The initial set of time series is divided into subsets or clusters, and then, each cluster is dealt with separately via cross-learning. The mechanism of action is that dividing the set can improve in-sample error more than the reduction in sample size impacts performance. Another typical example is removing 'outlier' time series from the set; for example, when the average training error is dominated by a few time series, we can consider removing them and treating them with local approaches. The extreme of clustering is completely separating the set, creating subsets formed by only one time series, this is in fact the local approach.
- **Joining datasets: Agglomeration.** Two sets of time series can be joined to form a single set that is cross-learned. For example, companies might have several forecasting tasks and might be approaching them separately, maybe even with cross-learning on each task, so another way of tuning performance could be to join the sets together.
- **Agglomeration can also be external, this means adding time series to the set that are not to be forecasted, purely to help performance.** This situation is not completely covered by the trade-off, because it relates to average performance across the series in the set, does not consider series outside of the dataset. However, in a practical situation, adding external time series to the set might not affect in-sample error by much. In these cases, it will likely improve the performance of a cross-learning model. The informal explanation is that adding external time series helped the training process by removing potentially poor solutions that could not be discarded with the original dataset. In practice, we believe that **careful external agglomeration will be the defining factor in performance of cross-learning**, so we encourage practitioner to consider it. One example can be expressing known properties of the time series in the dataset, when it is difficult to design a model with those properties. Take the property of "scale invariance", meaning that the scale of the series should not affect the forecasts (e.g. when we forecast monetary values, the currency used to express the value should not be relevant). Linear models are scale invariant, but we can make a

nonlinear model scale invariant by adding modified versions of the dataset at different scales.

- Fuzzy approaches can also be explored by practitioners, such as some time series appearing in more than one cluster.

## ILLUSTRATION OF CROSS-LEARNING APPROXIMATION OF SPECIFIC TIME SERIES PROCESSES

In this section, we will analyze specific examples of traditional time series processes such as growth curves, periodic patterns or stochastic processes, and see how the cross-learning paradigm applies to them. We start from known time series processes from a local perspective and show how to recover the exact equivalence from a cross-learning perspective. This is a single global predictive function that applies to several different time series processes, with different local predictive functions. The aim is to get a deep mathematical understanding of the cross-learning mechanism for some of the more relevant forecasting use-cases so that the reader will later adapt the principles to their own practice. We proceed “from the specific to the general”, as opposed to the results in sections “[Approximation Capacity of Cross-Learned Models](#)” and “[Statistical Trade-Offs](#)” that apply to general situations but give no explicit information about any particular process.

We use linear functions in our analysis because of their interpretability. Their simplicity facilitates insight, and by analyzing the limitations of linear approximations, we will motivate the use of Artificial Intelligence methods.

We develop the equivalence results in three steps:

1. First, we show examples of well-known families of time series processes and how they can be alternatively parameterized by local linear autoregressive functions. Section “[Well-Known Families of Time Series Processes Have an Equivalent Parameterization via Local Autoregression](#)”.
2. Second, we show how two different local linear autoregressive functions can be represented by a single cross-learned predictive function. With Steps 1 and 2, we already have an illustration of the main mechanism for cross-learning: from a representation

of two different local processes to a unique cross-learning representation that produces *exactly* the same forecasts. Discussed in section “[Two Different Linear Autoregressive Predictive Functions Can Be Represented by a Single Linear Autoregressive with More Lags](#)”.

3. Third, we show how an arbitrary number of local autoregressive processes can be represented by a single, global, cross-learning function either uses more lags than local and/or has nonlinear components. Discussed in section “[A Single Nonlinear Autoregression Can Represent an Arbitrary Number of Different Linear Autoregressions](#)”.

### *Well-Known Families of Time Series Processes Have an Equivalent Parameterization via Local Autoregression*

We will start by showing how many commonly appearing time series processes can be parameterized as linear autoregression. These processes are linear or quadratic growths, periodic functions such as sinusoidal or logistic curves. These processes, and particularly combinations of them, are behind many forecasting problems. A natural way to think about these processes is as a “function of time”; for example, a quadratic growth process is represented as:

$$f(t) = At^2 + Bt + C$$

with  $A, B, C$  the parameters. One approach to model this process from its observations is to impose a quadratic curve (e.g. based on plots or domain knowledge), fit its parameters  $A, B, C$  to data and then forecast by giving the recovered model values for time  $t$  that are outside the measured time points. We are going to see how we can also forecast these processes by modeling them as functions of past observations (autoregressive). The autoregressive parameterization serves two purposes:

1. It allow us to use a single model class to fit them. Parameterizations as a function of time require some knowledge of the underlying process: Is it quadratic? Exponential? Simple linear autoregression is more flexible; by tuning the parameters of an autoregressive model, we will be able to process a quadratic curve, cubic, exponential

growth, etc., there is no need to guess the type of curve, just tune the parameters of the autoregression.

2. The autoregressive parameterization will enable cross-learning across many different processes, shown in section “[Two Different Linear Autoregressive Predictive Functions Can Be Represented by a Single Linear Autoregressive with More Lags](#)”.

In our examples, we assume that a continuous time process is regularly sampled at some arbitrary frequency, for example at values of  $t$ ,  $t = 1, t = 2, t = 3, \dots$  producing an observed time series  $X_1 = f(1), X_2 = f(2), X_3 = f(3), \dots, X_t = f(t)$ . Then, we produce an autoregressive expression for the future value  $X_{t+1}$  as a function of past observations  $X_{t+1} = \phi_1 X_t + \phi_2 X_{t-1} + \dots + \phi_t X_t$ . We call the expression for  $X_{t+1}$  the forecasting equation; this expression has to be equivalent to the true process  $f(t+1)$  for any given value of  $t$ .

We show some examples of the autoregressive parameterization of some of the most popular processes.

- **Polynomials:** The equivalence between a polynomial curve and an autoregressive process can be shown via finite difference approximation. This is, differences between consecutive time steps of a time series approximate derivatives of the function that generates the process, and then, derivatives can be used to approximate any polynomial (we know this from Calculus). Polynomial processes are quite common in nature (linear or quadratic growth), but they can be used to approximate almost any true process if the polynomials are of high degree. As a consequence, to cover a wide variety of time series processes, it suffices to show that polynomial processes can be approximated by linear autoregressions. However, we will give examples of other notable families of processes to clarify the mechanism, and to show that some of these processes also have relatively simple representations as linear processes, without requiring to go through the “high degree polynomials approximate any curve” route. We will show two examples of polynomials, a specific quadratic curve and then the general procedure. Consider the generic quadratic curve:

$$f(t) = At^2 + B$$

Assuming  $t$  is the last observed time point, we can develop expressions for the future value  $f(t+1)$  by rewriting some of the observed time steps with respect to it:

$$f(t+1) = A(t+1)^2 + B = At^2 + B + At + A$$

$$f(t) = At^2 + B = f(t+1) - At - A$$

$$f(t-1) = A(t-1)^2 + B = At^2 + B - 2At + A = f(t+1) - 3At$$

$$f(t-2) = A(t-2)^2 + B = At^2 + B - 4At + 4A = f(t+1) - 5At + 3A$$

We observe that  $f(+1)$  can now be rewritten as a combination of  $f(t)$ ,  $f(t-1)$ ,  $f(t-2)$  *without any specific reference to t*, just purely autoregressive:

$$f(t+1) = 3f(t) - 3f(t-1) + f(t-2)$$

We can rewrite it as the observed time series  $X$  ( $f(t) = X_t$ ) so we get the forecasting equation:

$$X_{t+1} = 3X_t - 3X_{t-1} + X_{t-2}$$

which represents a linear autoregressive model with coefficients ( $\phi_1 = 3$ ,  $\phi_2 = -3$ ,  $\phi_3 = 1$ ). It can be tested that for any time series coming from that quadratic curve, the obtained forecast equation of the 3 more recent observations is able to produce exact forecasts.

The general principle follows from the classic result:

1. Taylor expansions: Functions can be approximated around a point by its derivatives:

$$f(t) = \frac{f'(a)(t-a)}{1!} + \frac{f''(a)(x-a)^2}{2!} + \frac{f'''(a)(x-a)^3}{3!} + \dots$$

2. Differences between consecutive observations of the time series can be used as an approximation of the derivatives of the functions: Any process  $f(t)$  as a function of time is observed

at time points  $f(t), f(t - 1), f(t - 2), \dots$ . The definition of derivative

$$f'(t) = \lim_{h \rightarrow 0} \frac{f(t) - f(t - h)}{h}$$

We plug  $h=1$ , and we recover an approximation of the derivative

$$f'(t) \approx \frac{f(t) - f(t - 1)}{1}$$

3. Plugging in the differences in place of the derivatives, what remains is a linear combination of the derivatives.

- **Periodic processes:** Any periodic process of period  $p$  follows the equation  $f(t) = f(t - p)$  by definition. Assuming that the period is an integer for simplicity, a time series that has been sampled over time, we would get the relationship  $X_t = X_{t-p}$ . This is in fact a linear autoregression model of order  $p$  with coefficient 1 at the most recent observed period and 0 elsewhere. We can write the forecasting equation:

$$X_{t+1} = 1X_{t-p+1} + \sum_{i=1}^{p-1} 0X_{t-i}$$

- **Exponential Growths/decays:** We can show that the expression for the next time step  $t + 1$  contains the expression for the last observation in our data  $t$ .

$$f(t) = Ae^{Bt}$$

$f(t+1) = Ae^{B(t+1)} = Ae^{Bt}e^B = e^B f(t)$  Therefore the forecast equation as that of an autoregressive process or order 1 with coefficient  $\phi_1 = e^B$ , this is:

$$X_{t+1} = e^B X_t$$

- **Logistic growth or sigmoid curves:** The family of sigmoid curves can be used to represent process that grows until “saturation”, describing an ‘S’ shape curve. They can be represented

mathematically in several ways, such as

$$f(t) = \frac{A}{1 + Be^{-C(t)}}$$

A transformation can be turned into a linear process:

$$g(t) = \log\left(\frac{A - f(t)}{f(t)}\right) = \log(B) - Ct$$

And then it can be easily modeled with a linear autoregressive model

$$g(t) = 2g(t - 1) - g(t - 2)$$

These three families can also be combined with each other to express a large variety of time series that are interesting for forecasting, though already only polynomials can be enough to approximate many real time series. The limitation of this equivalence is the “sampling frequency” and length of the time series. For example, to express a high degree polynomial, we will need to use many past observations for the high-order differences, more than the actual length of the series.

### *Two Different Linear Autoregressive Predictive Functions Can Be Represented by a Single Linear Autoregressive with More Lags*

We have shown that many typical processes that are naturally represented as functions of time can be also represented or parameterized as autoregressive functions. So far, this equivalence is “one to one” or local, in the sense that every different time series will have its own autoregressive representation. We now build up on this result and introduce the cross-learning equivalence; showing the “many to one” relationship, several different processes can be exactly predicted by a single autoregressive function.

To show the “many to one” relationship, we first illustrate the “two to one” relationship: from two different autoregressive processes, we show that there exists a third process that is able to express both of them. We will show the exact effect for a specific example and sketch the general underlying principle.

Suppose we have two different linear autoregressive processes of order 1:

$$X_{t+1} = \alpha X_t + \beta$$

$$Y_{t+1} = \gamma Y_t + \delta$$

If we give values for  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$ , we can find an autoregressive process of order 3 that can represent each of the two. For example, our two time series of interest are defined by:

- $\alpha = -2$   $\beta = -0.2$ ,  $\gamma = 1$  and  $\delta = -0.5$
- With some initial conditions  $X_0 = 1$ ,  $Y_0 = 0.3$

These values give rise to the two (very different) time series:

$$X = (X_0 = 1.0, X_1 = -2.2, X_3 = 4.2, X_4 = -8.6, X_5 = 17.0, \\ X_6 = -34.2, X_7 = 68.2, \dots)$$

$$Y = (Y_0 = 0.3, Y_1 = 0.8, Y_2 = 1.3, Y_3 = 1.8, Y_4 = 2.3, Y_5 = 2.8, \\ Y_6 = 3.3, Y_7 = 3.8, \dots)$$

It can be shown that the linear autoregressive function of order 3 with parameters ( $\phi_1 = 0$ ,  $\phi_2 = 3$ ,  $\phi_3 = -2$ ) can represent **both** X and Y. Therefore, both X and Y accept several parameterizations; one of those parameterizations is valid for both. Our starting parameterization was:

$$X_{t+1} = -2X_t - 0.2$$

$$X_{t+1} = Y_t + 0.5$$

but the following parameterization, applicable to both series, produces the same results:

$$X_{t+1} = 0X_t + 3X_{t-1} - 2X_{t-2}$$

$$Y_{t+1} = 0Y_t + 3Y_{t-1} - 2Y_{t-2}$$

Notice how this common parameterization uses more lags; it is a higher-order autoregression. In the scenario that requires fitting a model to data, this implies that the model class used for cross-learning is necessarily more complex (has more lags) than the model class that we would use for local modeling. On a general case for more than two time series, we can parameterize  $N$  several individual linear autoregressive functions of order  $L$  by a single linear of order at most  $NL + 1$ .

Adding lags to the global model has strong scalability issues; the more processes that we want to represent using a single autoregressive function, the more lags we will need, until in practical situations we do not have series of enough length. However, the limitation might not appear in practice, because the time series can be so similar that the common autoregressive parameterization will approximate well with a few more lags than local. The extreme case is when all processes have the same parameters or very similar; then, the global parameterization will not need extra lags. There is strong empirical evidence for this effect even for very large datasets. For example, thousands of monthly time series in the M3 dataset are represented well enough (better than local traditional models) by a single autoregressive function of high order (Montero-Manso & Hyndman, 2021).

More importantly, in the general sense, the limitation of having to add a large number of lags to the model is resolved by adding “variable transformations”: a global nonlinear process of a few lags will represent an infinite number of local linear processes (see section “[A Single Nonlinear Autoregression Can Represent an Arbitrary Number of Different Linear Autoregressions](#)”).

### **Guidelines for cross-learning derived from adding lags to global models**

Start from an autoregressive model class that is appropriate for local modeling of the time series, an “initial guess” and try global modeling using that class. Add lags to the model sequentially and validate the performance. For example, we might know or have some background information that a monthly time series has some seasonal pattern, so we could impose a local model of lag 12 or some multiple of 12. For global modeling, we could start from those 12 lags and then add lags to the model one by one. The effect of lags on performance has been experimentally shown to be non-monotonic, so it is possible that adding one lag decreases performance, but adding two lags increases performance again; therefore, some extra lags after an “optimal” is found should be checked

to make sure that we are not stopping in a local minimum. The idea of adding lags can be applied to nonlinear autoregressive classes, and we can think of adding lags as “linearizing” the model; therefore, it can make a nonlinear (e.g. neural network) model more stable or better at extrapolation. The effect of nonlinearities for local and global equivalence is discussed in section “[A Single Nonlinear Autoregression Can Represent an Arbitrary Number of Different Linear Autoregressions](#)”.

Even though the theoretical result indicated that we need to add many lags to guarantee equal approximation to local models, realistic applications might not require many lags, for example when the processes are “similar” or were preprocessed to become more similar. For example, when forecasting growth patterns, normalization and time aligning the processes so they start growing at similar time points can make the processes become very similar.

### *A Single Nonlinear Autoregression Can Represent an Arbitrary Number of Different Linear Autoregressions*

There is a way of representing an **arbitrary** number of linear autoregressive functions as a single autoregressive function that does not require as many lags. We can achieve this by introducing nonlinearities in the autoregression; this is, a single **nonlinear** autoregressive function with a few more lags than the more complex linear local representation can approximate a set of series of arbitrary size.

We will find this nonlinear function by showing how a linear process of order 1 also has a nonlinear representation, and then, we realize that this nonlinear representation is actually “universal”; it is valid for all linear processes of order 1.

We start from our generic linear autoregressive process:

$$X_{t+1} = \alpha X_t + \beta$$

which we can unroll one time step:

$$X_{t+2} = \alpha X_{t+1} + \beta$$

These two expressions are actually equivalent, the second is just one time step “unrolled” from the first (the more common notation), since the value of  $t$  is arbitrary.

What we achieve with this unrolling is that we now have two representations of the same time series, or two equations for the same process. With these two equations, we can solve for  $\alpha$  and  $\beta$ , this is, find a way of expressing  $\alpha$  and  $\beta$  as a function of the observations of the time series. Once  $\alpha$  and  $\beta$  are removed, the resulting expression will be valid for **all** processes of order 1, all possible values of  $\alpha$  and  $\beta$ !

Solving for  $\alpha$  and  $\beta$ , assuming no division by zero:

$$\alpha = \frac{X_{t+1} - X_{t+2}}{X_t - X_{t+1}}$$

$$\beta = \frac{X_t X_{t+2} - X_{t+1}^2}{X_t - X_{t+1}}$$

We now unroll the process one more time

$$X_{t+3} = \alpha X_{t+2} + \beta$$

and plug in solutions for  $\alpha$  and  $\beta$ , resulting in:

$$X_{t+3} = \frac{X_{t+1} X_{t+2} - X_{t+2}^2 + X_t X_{t+2} - X_{t+1}^2}{X_t - X_{t+1}}$$

We can see that this is a nonlinear autoregressive function. For clarity, we can rewrite the expression so that the prediction is for  $t + 1$  instead of  $t + 3$ , since it does not matter which  $t$  we use. Therefore, we show that any number linear autoregressive process of lag 1 can be expressed as a nonlinear autoregressive process of lag 3.

$$X_{t+1} = \frac{X_{t-1} X_t - X_t^2 + X_{t-2} X_t - X_{t-1}^2}{X_{t-2} - X_{t-1}}$$

Plugging in this formula for any linear autoregressive process or order 1 (such as the examples of section “[Two Different Linear Autoregressive Predictive Functions Can Be Represented by a Single Linear Autoregressive with More Lags](#)”) will perfectly reproduce the time series. The exception to this formula is the constant time series that generates a division by 0 in the formula, though these time series are trivial.

We can generalize the principle to state (without proof) that single cross-learned nonlinear autoregressive functions are able to express

complete families of the most local common time series processes. The more complex the individual processes, the more lags and nonlinearities will be required. Moreover, there might be more than one common representation, and some model classes might be more suitable for practice than others. For example, using as model class a deep network can also produce the common parameterization, the logic for this claim being that if a “polynomial” process can represent an arbitrary number of series, since a deep network with enough layers and units can approximate a polynomial, then a deep network can also represent an arbitrary number of local processes.

**Guidelines derived from the existence of nonlinear cross-learned representations applicable for an arbitrary number of different local processes**

We have shown how a single global nonlinear model can represent an arbitrary number of local linear processes. Not only in number of series, but the local process can be very different, as in their local parameters are different or the series look different when plotted, such as exponential trends, linear trends, seasonal, etc. All are covered by a single nonlinear cross-learned model, because all underlying local processes are linear. The recommendation is more of a motivation: cross-learned nonlinear autoregressive processes can succeed in capturing the arbitrary sets of time series that come from local processes of the kind that we most often encounter in practice. If we combine this approximation result with the statistical trade-off, we get the insight that model complexity for cross-learned does not really need to be that much higher. In the example of this section, we just add some “simple polynomial transformations”, just a few more parameters, and it might capture an arbitrary number of local processes (millions...), so the trade-off of the cross-learning can be extremely beneficial: we get a much larger sample size for a small increase in complexity.

### *Cross-Learning for Processes with Noise or Perturbations*

We have shown how a single cross-learned function can be applied to predict many different local linear processes that are exact or “deterministic”. While this result covers natural applications of time series forecasting, the equivalence is for “idealized” processes, and most realistic applications will have a “noise component” or a perturbation that is complex enough

to be considered noise. In this section, we provide a cross-learning equivalence for the most simple cases of processes with noise, which is a time series that is generated by a constant plus noise.

Suppose that we have a set of  $N$  time series of length  $T$  that can be described as “constant plus noise” processes. All these time series have different constants and noise distributions. The noise processes can be considered independent and identically distributed observations generated from a 0-mean probability distribution; the distribution function and variance can vary across series. The  $i$ -eth series in the set can be represented as:

$$X_{it} = C_i + \varepsilon_{it}$$

In the local approach, we can forecast these time series by computing the sample mean of each time series (if we are interested in a prediction that minimizes the squared error). There is a global or cross-learned autoregressive function that produces equivalent predictions to the local approach. It is the linear autoregressive function of order  $T$  (the complete length of the time series) with coefficients  $\frac{1}{T}$ ,  $(\phi_1 = \frac{1}{T}, \phi_2 = \frac{1}{T}, \dots, \phi_T = \frac{1}{T})$ :

$$X_{it+1} = \frac{1}{T} \sum_t^t X_{it}$$

This autoregressive function actually computes the sample mean of each series; we can view the formula for the sample mean that we compute locally as a cross-learned linear autoregressive function that we apply to all time series. This simple example illustrates the statement in section “[Approximation Capacity of Cross-Learned Models](#)” for estimation. The local modeling applies the same function to all time series, the sample mean. The global modeling applies the same function, but this function is learned from the data, and in this case can quickly converge to the “optimal” local function. Moreover, in realistic scenarios, the true underlying local model class is not known and some form of model selection that considering complex models is applied. Model selection usually incurs a cost in prediction error compared to starting from the true model class. In our example of the constant plus noise processes, we could think that the true model could be an autoregressive linear of order 0, so only an additive constant or intercept is fit. However, when the nature of the

processes is not known, we would be applying a complex model such as an ARIMA or exponential smoothing. These models can reduce to the sample mean via estimating their parameters, but will likely perform worse than the AR(0) for realistic sample sizes. However, the global approach will learn that the true model is the sample mean by considering all time series in the dataset. Meanwhile, the prediction error of a local approach would not scale at all with the size of the set. A cross-learning approach, however, could likely find from data values for its coefficients that quickly approximate the sample mean. Reasoning about how similar the constants of the true processes are (not shown here) would lead to more insights comparing local and cross-learned methods.

#### **Guidelines derived from the analysis of noisy local processes**

The main recommendation derived from the analysis of process plus noise is a reinforcement of one of the main principles of cross-learning established before: model classes for cross-learning have higher-order autoregressions than what we normally consider reasonable for local approaches.

- **In the case of noisy processes, a large order autoregression is necessary for the cross-learned model class.** The lags are needed to average across observations, to ‘smooth out’ the noise. We highlight the term “necessary” because we can make the processes approximate well by adding nonlinearities instead of lags. However, in the case of noise, nonlinearities are not sufficient. This recommendation might contrast with some existing rules of thumb for cross-learning models, such as trying autoregressions with lags up to two seasons (e.g. lag 24 when the series are monthly with yearly cycles).
- **“Sparsity” might not be helpful for cross-learning in the presence of noise.** By ‘sparsity’, we mean Machine Learning/Artificial Intelligence techniques with the purpose of making the model focus on a set of a few input variables, ignoring the rest, for example via L1 or L0 regularization, simple feature selection via model selection or by carefully crafted deep networks that induce this property. The reason follows from the same principle, we need to average out the error across many lags, so we should not be removing lags from the model.

## EXPLICIT EXAMPLE OF TRANSFER LEARNING

Until now, we have established the possibility of positive statistical trade-offs as the main motivation for cross-learning. From a statistical point of view, the predictive performance of cross-learning scales with the size of the set of time series, while traditional local models do not. Cross-learning, however, requires more complex model classes than local, so they are more difficult to fit right, and hence, we get a trade-off.

In this section, we illustrate another benefit of cross-learning, the possibility of automatic “transfer of information” across time series that goes beyond the standard statistical trade-off paradigm of sample size vs model complexity. We categorize this example under the umbrella term ‘transfer learning’, as it refers to how solving one problem (one time series) can improve performance in other problems (other time series).

It can be argued that practitioners are constantly doing transfer learning when forecasting. For example, suppose we have been forecasting time series of book sales for years. It is very likely that when we face the problem of forecasting the sales of a new book, we manually exploit some of the knowledge gained from forecasting other books, and we leverage our experience gained. This knowledge can appear in several ways:

- We already know which model classes work well for other books (and which models don’t do well), refining our model selection. When forecasting a new time series, we do not try models that never worked before.
- We might have come up with completely new models for the task at hand.
- We have some knowledge of which variable transformations or preprocessing steps are suitable for this task.
- We can even get specific values for some of the parameters of the models so we do not fit them to the data of the new time series, and we just set them. This is common practice when we have very little historical data for the new book.

We emphasize that this type of transfer is applied “manually” or subjectively; in a non-systematic fashion, it is strongly dependent on the practitioner expertise. Our illustration will show how cross-learning can do this transfer automatically. We will use the real case of COVID-19 forecasting in the early stages of the pandemic in Europe as context.

In this setting of COVID-19 predictions, particularly in the early stages, many practitioners used “information transfer” to make better predictions. For example, the Institute for Health Metrics and Evaluation (IHME) used epidemiological models (a sigmoid curve model) to make predictions for the United States, but some of the parameters for these curves were estimated from data from outside the United States, from countries that had already peaked such as China, Italy or Spain (IHME COVID-19 Health Service Utilization Forecasting Team & Murray, 2020). A similar example can be found in Lee et al. (2020). Informally, they argue that it is more reliable to estimate some of the parameters of the model (e.g. mortality rate) on countries that have already peaked, and then use these parameters to forecast countries that are still on the earlier stages of the pandemic (pre-peak). Before the peak, countries have less data and estimating parameters in the growth phase of the curve is very unstable.

These are examples of manual transfer of information, as they require the practitioner to:

- Have deep expertise in the field, imposing the right models (a particular kind of sigmoid) with the right parameterization.
- Know which of these parameters can be shared or transferred from some series to the others. Mortality rate might be more similar across countries, while infection rate might vary based on very different population densities, infrastructure, lifestyle, etc.
- Know the appropriate way to transfer the parameters when they require adjustment or transformations. For example, we would need to adjust mortality rate based on the age distribution differences in each of the countries involved.
- Have understanding about the pre-peak/post-peak behavior, to know that post-peak series are more reliable sources of information for estimating the parameters.

Needless to say, this kind of knowledge is not always available.

In the case of Europe, Italy was a few days “ahead” into the pandemic compared to other European countries, and they also introduced interventions such as lock-downs before other countries (e.g. Italy introduced lock-downs on 9 March 2020, UK on 28 March 2020). Therefore, the

availability of data from Italy (daily time series of new cases, deaths, hospitalizations) might help forecast other countries: if we want to make a prediction about the impact of lock-downs in the UK in late March 2020, Italy is already two weeks into its own lock-down and we can learn a lot by observing them. While the assumption of having an intuitive understanding that Italy is ahead of other countries and can “help” forecast them is realistic, knowing how to do this efficiently is far from a solved problem. However, from the cross-learning point of view, we can make a very strong case that this property of transfer of information appears automatically. We make a three-step argument:

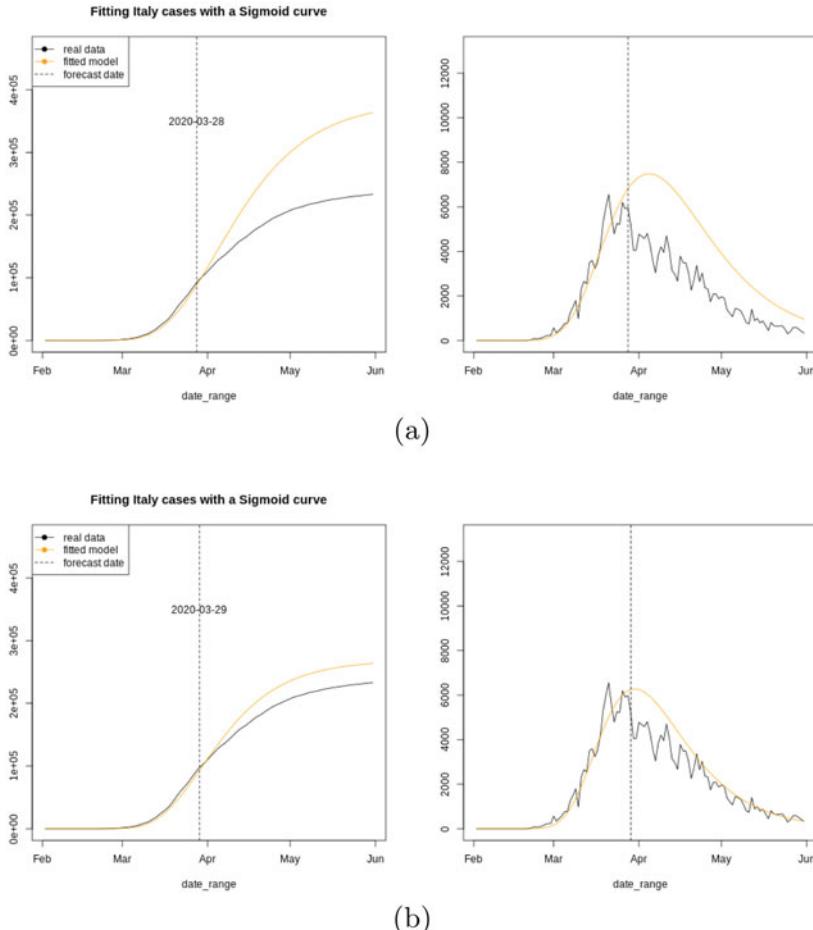
1. Cross-learning finds a single predictive function for a whole set of series that fits the data as well as local models. We can get the same predictions with cross-learning than with local, we have even established the mechanism for this in sigmoid curves in section “[Well-Known Families of Time Series Processes Have an Equivalent Parameterization via Local Autoregression](#)”.
2. Some time series in the set have information that can be transferred to others. We call the time series with the information “source” time series, and the time series that can benefit from the information are the “target” time series. This information in the source series does not need to be useful to themselves; it might only apply to the target series.
3. Fitting a cross-learning function to the source time series means extracting the information from them. Since the cross-learned function is also a good fit for the target time series (which are also in the set), then cross-learning is transferring the information from source to target. From another point of view, finding a good global model for the source time series means discarding “bad” candidate models for the source time series. These bad candidate models would not be discarded if our dataset consists only of target time series.

We now give a visual representation of the transfer of information effect via cross-learning and its potential impact in accuracy. We use Italy as the source time series and the United Kingdom as the target series. This is a real data example but selected a posteriori for the purposes of illustration. However, this principle of transfer learning via cross-learning was actually applied in real time for making COVID-19 predictions as part

of ensemble forecasts in Spain (using Italy as source), Australia (using several other countries as source) and for the European Union (Sherratt et al., 2023). The performance of this approach was on par with the best models used by experts, and it really shined on the very early stages of the pandemic for forecasting complex variables such as ICU beds occupation.

We will go to the 28th and 29th of March 2020, when it can be argued that the cases for Italy ‘peaked’. Figure 6.1 shows the predictions for infections of a fundamental (based on epidemiology) sigmoid curve model fitted on the 28th (top figure) an 29th (bottom figure) of March 2020, using orange color. Vertical striped lines in the figures represent the forecast day; this is, the models are fit to all observation up to that date. By looking at the actual data (solid black lines), we can see that infections peaked around those dates, and there is a substantial difference in performance between the prediction on the 28th and the 29th. Predictions on the 28th are quite poor, and predictions made *any day* after the 28th are much better. Therefore, we consider that from the 29th, Italy is a reliable source of information for how the pandemic evolves.

Imagine now that we want to make predictions for infections in the United Kingdom on the 29th of March 2020. While Italy has already peaked and its predictions are very reliable, the infections for the UK have clearly not yet peaked (Fig. 6.2), and therefore, the sigmoid model applied to the UK data on the 29th is not very accurate (Fig. 6.2a). Experts could try to manually “transfer” some of the information about the curve from Italy to the UK based on the sigmoid model. For example, we could use the values for the parameter that controls the growth rate of the curve estimated in Italy for the UK curve. Other parameter to transfer could be “days until the peak” assuming that UK will take the same amount of time to peak since the first case was registered, maybe adjusting by delays in the establishment of lock-downs, etc. These two examples of transfer imply that we ignore what the data from the UK is telling us. For example, instead of using the parameters that best fit the UK data, we use the parameters that best fit in Italy (maybe with an additional manual adjustment). Another approach is to try the generic cross-learning via autoregressive models instead of the epidemiologically motivated sigmoids. Figures 6.2b and 6.2c show the effect of the cross-learning approach to this problem. Figure 6.2b shows the predictions of the best-fit autoregressive model applied to the UK locally (using only the UK data, not cross-learning). It is obvious that the performance of local autoregressive is also very poor, even worse than the sigmoid, since an



**Fig. 6.1** **a** Predictions of a sigmoid model fit to data for the COVID-19 infections in Italy using data up to 28 March 2020. The vertical striped line indicates the date on which the forecast is made, and the model is fit using only data before that date. Left panel represents daily cumulative cases, and right panel represents new infections per day. Long-term predictions are very off the mark, arguably because the models are very unreliable before the peak of the series. **b** Predictions of the same model, based on data on the following day, 29 March 2020. Predictions are much more accurate, and we can identify the pre-peak/post-peak behavior of the model. Fitting the same model on subsequent dates also produces reliable predictions.

autoregressive model is much more complex than the sigmoid: it uses more parameters; if we let it fit freely to data, it can produce totally unrealistic pandemic curves, such as negative values.

Figure 6.2c shows the predictions of an autoregressive model that has been cross-learned using data from the UK grouped with data from Italy. This means that predictions come from the best-fit model in a least squares sense, the model that works best in the data for *both* UK and Italy. The cross-learned model has much better performance than either the sigmoid or local autoregressive applied to the UK. The performance of the cross-learned model is on par with the post-peak sigmoid model applied to Italy. Therefore, we conclude that the post-peak information available in Italy has been successfully transferred to the UK via cross-learning. The main benefit of the cross-learning approach is that this transfer happens automatically, and we get the benefits just fitting models to data in a straightforward way. It can even be argued that the intuitive notion that “Italy has some information that can benefit other countries because it is some days ahead of them in the pandemic” is not even required. In cross-learning, we pool data together “by default”, so it is very likely that when forecasting pandemic curves, we pool together data from all possible countries, and transfer learning happens “for free” or unknowingly. Of course, this does not mean that we should not curate the data that we feed to our models.

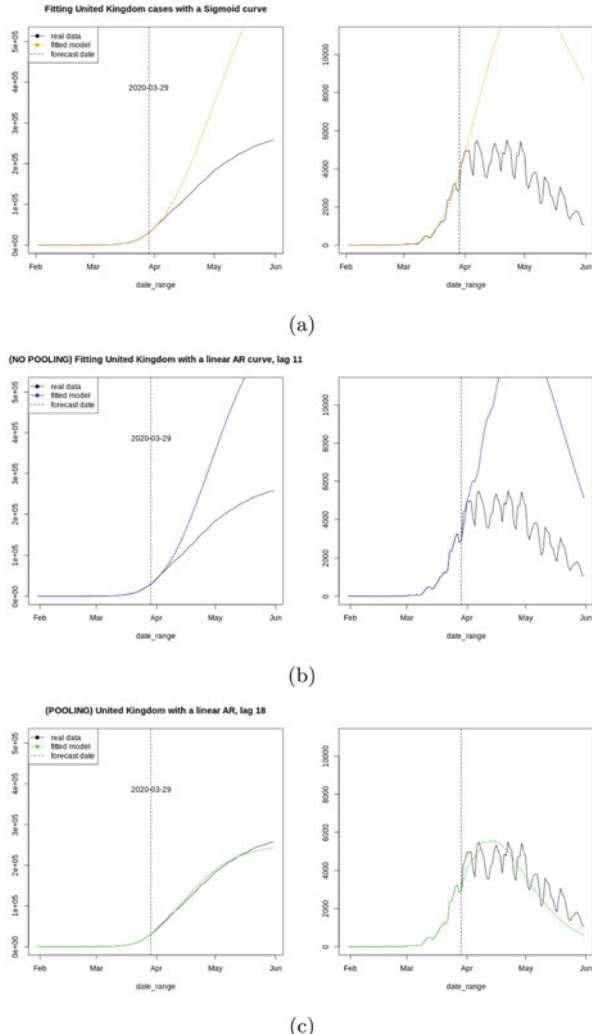
### **Guidelines for Transfer Learning**

The practical recommendation coming from the COVID-19 illustration is a special case of the general recommendation from section “[Statistical Trade-Offs](#)” that suggests including time series in the set for which we are not interested in their forecasts; they are only there to improve performance of the series that we care about.

We recommend adding to the “set of time series of interest” some external “source” time series that can be considered to be “ahead of time” from the perspective of the series of interest.

Examples of potential “sources time series” are:

- Products that have been launched in some areas before others.
- Periodic patterns already fully observed in some series and not in others.
- Effects of interventions (policy changes) that have been applied to some processes before others.



**Fig. 6.2** **a** Predictions of a sigmoid model fit to data for the COVID-19 infections in the UK using data up to 29 March 2020, when Italy had already peaked. Since the UK has not peaked, the predictions of the sigmoid model are not accurate. **b** Predictions of a local (not cross-learned) linear autoregressive model for the UK using the same data. The performance of a local linear autoregressive model is even more inaccurate than those of the more fundamental sigmoid model, since the **c** predictions for a cross-learned linear autoregressive model that includes the time series for Italy in the set. The cross-learned model has effectively transferred the pre-peak/post-peak behavior from Italy to the UK.

- Natural geographic differences such as summer happening on the southern hemisphere before the northern every calendar year.
- Cultural differences such as the main month for summer holiday period varying across countries (e.g. July vs August).

## CONCLUSIONS

We have shown an analysis of the cross-learning mechanism for time series forecasting, a method that pools data together from a dataset of multiple time series to fit a single function that is used to forecast all time series in the dataset.

The main consequence for Artificial Intelligence applications to forecasting is that once we decide to make the leap from the traditional time series models to Artificial Intelligence, we should also abandon the paradigm of fitting those models to each time series in isolation. We should try to train AI models across multiple time series, even when we are only interested in forecasting one of them. The recommendation could be: “Cross-learn Artificial Intelligence models and actively tune the dataset of series that is used for training, consider the dataset as another parameter”. Cross-learning is supported throughout this chapter, however local, individual analyses can help guide the cross-learning modeling, such as when would we use cross-learning, which models could work, which time series should be grouped together, what are the reasons why cross-learning works so well, etc.

Our analyses show that cross-learning is applicable to arbitrary datasets of any size and diversity, in the sense that cross-learning can always produce the same forecasts than traditional local approaches that fit and predict each time series in isolation. This expands the application of cross-learning from its intuitive use case of datasets of homogeneous/similar time series to any forecasting setting. However, in large and diverse datasets, the types of models that should be used for cross-learning have to be different from the models commonly used for individual time series. In particular, cross-learning models should be more complex than classical time series models, motivating the use of main Artificial Intelligence models such Deep Networks or Decision Trees.

A second analysis sheds light into the statistical performance of cross-learning compared to local models, showing that for comparable model classes, able to fit the data well, the performance of cross-learning will

improve with the size of the set compared to local learning. Therefore, not only cross-learning is applicable to large datasets, it will also improve its performance.

The main limitation of cross-learning lies in the difficulty of finding good model classes, as they have to be much more complex than the local models, and domain knowledge is much more difficult to express in complex models. To mitigate this effect, we derive the recommendation adding to our dataset “external” time series that show the properties that we would like to express in our models. These time series could even come from simulations of idealized processes.

We also analyze specific examples of commonly appearing time series processes, such as exponential growth, sigmoids, periodic or polynomial trends to see how cross-learning models . In other words, we derive exact solutions to the ideal cross-learning model when the ideal local model is known. Two key insights from this analysis are derived. First, autoregressive cross-learning models must be of higher order than local (have more lags). Second, while the analysis uses linear autoregressive models, it also shows how nonlinear models are necessary for successful cross-learning for scalability purposes and how nonlinearities can even lead to a better statistical trade-off than linear models. We show that there might be multiple nonlinear approximation for the same dataset.

Even though we cover a few examples instead of providing general results, there is value in the exposition and we recommend the reader to apply a similar thought process for their particular forecasting tasks: try to express some idealized fundamental process from your task and then derive the cross-learning equivalence using the tools we showcased. This could lead to model classes that are better suited to that problem, as opposed to generic AI models.

Finally, we provided an example for an additional benefit of cross-learning that is difficult to explain from the traditional statistical viewpoint: cross-learning can automatically transfer information from some series to others, critically improving the forecasting accuracy beyond what is possible with local learning. An illustration for this transfer learning effect comes from COVID-19 forecasting, when the post-peak dynamics of Italy are transferred to other time series, such as the UK, to improve their forecasting under pre-peak uncertain conditions, greatly outperforming fundamental epidemiological models and local autoregressive models that forecast times series in isolation.

## REFERENCES

- Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, 16(4), 521–530.
- Godahewa, R. W., Bergmeir, C., Webb, G. I., Hyndman, R., & Montero-Manso, P. (2021). Monash time series forecasting archive. In *Thirty-fifth conference on Neural information processing systems datasets and benchmarks track*.
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.
- IHME COVID-19 Health Service Utilization Forecasting Team & Murray, C. J. L. (2020). Forecasting the impact of the first wave of the COVID-19 pandemic on hospital demand and deaths for the USA and European Economic Area countries. *MedRxiv* (pp. 1–15).
- Lee, S. Y., Lei, B., & Mallick, B. (2020). Estimation of COVID-19 spread curves integrating global data and borrowing information. *PLoS ONE*, 15(7), e0236860.
- Makridakis, S., & Hibon, M. (2000). The M3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4), 451–476.
- Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4), 1632–1653.
- Rajapaksha, D., Bergmeir, C., & Hyndman, R. J. (2022). LoMEF: A framework to produce local explanations for global model time series forecasts. *International Journal of Forecasting*, 39(3), 1424–1447.
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.
- Sherratt, K., Gruson, H., Grah, R., Johnson, H., Niehus, R., Prasse, B., Sandman, F., Deuschel, J., Wolffram, D., Abbott, S., et al. (2023). Predictive performance of multi-model ensemble forecasts of COVID-19 across European nations. *Elife* (p. 12).



## CHAPTER 7

---

# Handling Concept Drift in Global Time Series Forecasting

*Ziyi Liu, Rakshitha Godahewa, Kasun Bandara,  
and Christoph Bergmeir*

## INTRODUCTION

Accurate time series forecasting is crucial in the context of decision-making and strategic planning nowadays for many businesses and industries. The forecasting models are typically trained based on historical data. In most applications, data distributions are not stationary and they change over time while making the trained models outdated and reducing their forecasting accuracy, which is known as *concept drift*. Hence, it is important to make sure that the trained models well-generalise beyond the training data and are capable to provide accurate forecasts even with the changed data distributions. Transfer learning can be applied

---

Z. Liu · R. Godahewa · C. Bergmeir (✉)

Department of Data Science and AI, Monash University, Melbourne, VIC, Australia

e-mail: [christoph.bergmeir@monash.edu](mailto:christoph.bergmeir@monash.edu)

Z. Liu

e-mail: [ziyi.liu1@monash.edu](mailto:ziyi.liu1@monash.edu)

to make the models robust to distribution shift. In general, transfer learning is an approach of solving one problem and applying the corresponding knowledge gained to solve another related problem. In the time series forecasting domain, the transfer learning models typically train on a large set of series and during forecasting, they predict the future of (1) new series that they have not seen before; (2) training series where the data distributions change over time. In this chapter, we focus on the second sub-problem of transfer learning which is predicting the future of a set of series that is used to train a model when the corresponding data distribution is not stationary.

Traditional forecasting methods like Exponential Smoothing (ETS, Hyndman et al., 2008) are capable of handling the most common non-stationarities, namely trends and seasonalities, and also have been designed to show a certain degree of robustness under other types of non-stationarities. In particular, ETS gives most weight to the most recent observations, and therefore, it can adapt to a distribution shift reasonably well. However, Global Forecasting Models (GFM, Januschowski et al., 2020) have recently shown their potential in providing more accurate forecasts compared to the traditional univariate forecasting models, by winning the M4 (Makridakis et al., 2018) and M5 (Makridakis et al., 2022) forecasting competitions. In contrast to traditional univariate forecasting models that build isolated models on each series, GFMs build a single model across many series while allowing the model to learn the cross-series information. By building a single model over many series, the models have more data available for training and can afford to be more complex (Montero-Manso and Hyndman, 2021). Consequently, ML methods like neural networks and Gradient Boosted Trees are usually

---

R. Godahewa

e-mail: [rakshitha.godahewa@monash.edu](mailto:rakshitha.godahewa@monash.edu)

K. Bandara

School of Computing and Information Systems, University of Melbourne,  
Melbourne, VIC, Australia

e-mail: [kasun.bandara@unimelb.edu.au](mailto:kasun.bandara@unimelb.edu.au)

C. Bergmeir

Department of Computer Science and Artificial Intelligence, University of  
Granada, Granada, Spain

the methods of choice for global modelling. However, unlike many traditional forecasting methods like ETS, ML methods do normally not have inherent mechanisms to deal with changes in the underlying distribution. Therefore, in ML the field of concept drift has emerged, which studies how ML methods can be used under distribution shift (Webb et al., 2016). In this chapter, we study how the methods from the concept drift literature can be used for forecasting with ML methods.

To the best of our knowledge, no methods to deal with concept drift in the forecasting domain in a GFM and ML context have been proposed in the literature. The available concept drift handling methods are all related to the classification domain. Dynamic Weighted Majority (DWM, Kolter & Maloof, 2007), Social Adaptive Ensemble (SAE, Gomes et al., 2013) and Error Intersection Approach (EIA, Baier et al., 2020) are some of the popular concept drift handling methods in the classification domain. Most of these concept drift handling methods use ensembling (Gomes et al., 2017). In the forecasting space, ensembling is also known as forecast combination which aggregates the predictions of multiple forecasting models to produce the final forecasts. Ensembling techniques are widely used to reduce model variance and model bias. This motivates us to explore methods to handle concept drift in the GFM space that incorporate ensembling techniques.

In this study, we propose two new forecast combination methods to handle the concept drift in the forecasting domain. We name these two methods Error Contribution Weighting (ECW) and Gradient Descent Weighting (GDW). In particular, these methods independently train two forecasting models based on different time series, typically one model is trained with the most recent series history and the other model is trained with the full series history, where the two models are finally weighted based on the previous prediction error, which is known as continuous adaptive weighting. The proposed ECW and GDW methods are different based on the method of calculation of the sub-model weights. In the proposed methods, the sub-model trained with the full series' history is expected to provide more accurate forecasts in general situations whereas if concept drift occurs, then the sub-model trained with the most recent series' history is expected to quickly adapt to the new concept. Furthermore, two different methods for series weighting, exponential weighting and linear weighting, are used during the training of the two sub-models where these methods internally assign higher weights for the most recent series observations than the older observations during training. We also

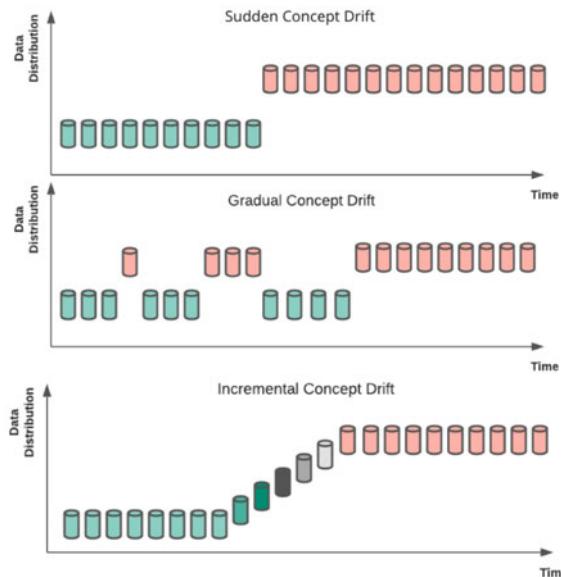
quantify the effect of handling concept drift using our proposed methods and only using the series weighted methods considering LightGBM (Ke et al., 2017) as the base learner. The LightGBM models that use our proposed concept drift handling methods outperform a set of statistical benchmarks and LightGBM baselines with statistical significance across three simulated datasets on four error metrics. All implementations of this study are publicly available at: [https://github.com/Neal-Liu-Ziyi/Concept\\_Drift\\_Handling](https://github.com/Neal-Liu-Ziyi/Concept_Drift_Handling).

The remainder of this chapter is organised as follows: Section “[Problem Statement: Concept Drift Types](#)” introduces different concept drift types. Section “[Related Work](#)” reviews the relevant prior work. Section “[Methodology](#)” explains the series weighted methods and proposed concept drift handling methods in detail. Section “[Experimental Framework](#)” explains the experimental framework, including the datasets, error metrics, benchmarks, evaluation and statistical testing. An analysis of the results is presented in Section “[Results and Discussion](#)”. Section “[Conclusions and Future Research](#)” concludes the study and discusses possible future research.

## PROBLEM STATEMENT: CONCEPT DRIFT TYPES

There are two main types of concept drift: real concept drift (Kolter & Maloof, 2007) and virtual concept drift (Delany et al., 2004). Real concept drift occurs when the true outputs of the instances change over time whereas virtual concept drift occurs when the data distribution changes over time even with the same true outputs, possibly due to noise. In many situations, real and virtual concept drift can co-occur.

The literature typically classifies real concept drift into four different groups (Webb et al., 2016) which are (1) sudden concept drift where the data distribution at time  $t$  suddenly changes into a new distribution at time  $t + 1$ ; (2) incremental concept drift where the data distribution changes and stays in a new distribution after going through a variable distribution; (3) gradual concept drift where the data distribution shifts between a new distribution and the current distribution over time where gradually, the series completely enters the new distribution; (4) recurring concept drift where the same concept drift occurs periodically. Incremental and recurring concept drift in a time series forecasting context



**Fig. 7.1** A visualisation of sudden, gradual and incremental concept drift types

is closely related to trends and seasonalities in the data. In particular, recurring concept drift can be easily handled by the typical handling of seasonality with dummy variables, Fourier terms and similar additional features (Hyndman and Athanasopoulos, 2018). Thus, in this study, we focus on three concept drift types: sudden, incremental and gradual, which are illustrated in Figure 7.1.

## RELATED WORK

In the following, we discuss the related prior work in the areas of global time series forecasting, transfer learning and concept drift handling.

### *Global Time Series Forecasting*

GFM (Januschowski et al., 2020) build a single model across many series with a set of global parameters that are the same across all series. In contrast to the traditional univariate forecasting models, GFM are

capable of learning cross-series information with a fewer amount of parameters. Global models have been pioneered by works, such as Salinas et al. (2020), Smyl (2020), Bandara et al. (2020), Hewamalage et al. (2020) and Godahewa et al. (2021).

GFM have recently obtained massive popularity in the forecasting field after winning the M4 (Makridakis et al., 2018) and M5 (Makridakis et al., 2022) forecasting competitions. In particular, the winning method of the M4 competition uses global Recurrent Neural Networks (RNN, Hewamalage et al., 2020) whereas the winning method of the M5 competition uses global LightGBM (Ke et al., 2017) models.

### *Transfer Learning*

In the forecasting domain, there are many works that present on forecasting new time series that a model has not seen during training. Oreshkin et al. (2021) propose a deep learning-based meta-learning framework that generalises well on new time series from different datasets that it has not seen before. The popular Nixtla framework (Canseco & Garza, 2022) provides ten N-BEATS (Oreshkin et al., 2020) and N-HiTS (Challu et al., 2023) models that are pre-trained using the M4 dataset (Makridakis et al., 2018) where the models are expected to generalise on unseen time series. Woo et al. (2022) propose DeepTime, a meta-learning framework that can forecast unseen time series by properly addressing distribution shifts using a ridge regressor. Grazzi et al. (2021) propose Meta-GLAR, which performs transfer learning using local closed-form adaptation updates. The transfer learning framework proposed by Ye and Dai (2018) in particular considers the adequate long-ago data during training.

To the best of our knowledge, all these transfer learning frameworks produce forecasts for unseen time series. In contrast, our proposed methods produce forecasts for the same set of time series used during training where the distribution of the series is not stationary.

### *Concept Drift Handling*

To the best of our knowledge, the existing concept drift handling methods are all related to the classification domain. The concept drift handling methods in the literature can be mainly divided into two categories: implicit methods and explicit methods (Ghomeshi

et al., 2019). The explicit methods use a drift detection technique and immediately react to the drift at the point of its occurrence. The implicit methods do not use drift detection techniques.

In particular, Chu and Zaniolo (2004) propose the Adaptive Boosting Model which is an explicit method of concept drift handling in the classification domain. This method assigns a weight to each training instance where the weights are consequently updated across a sequence of classifiers. The first classifier assigns a sample weight to each instance. Each classifier then keeps the same weights for the correctly classified instances and assigns higher weights for the incorrectly classified instances where the weighted instances are used to train the next classifier in the sequence. Each classifier is also assigned a separate weight based on its classification accuracy. This process is repeated until the error function does not change. The weighted average of the predictions provided by all classifiers is considered as the final prediction. When a concept drift situation is detected, the weight of each classifier in the sequence is reset to one. However, this method contains a few limitations. It is highly time-consuming for trivial concept drift. Also, if the concept drift is too small to impact the accuracy of the previous classifier, it may stop building new classifiers. Furthermore, it needs to reset all classifiers when a data stream encounters concept drift which may then lead the model to be sensitive to false alarms (Krawczyk et al., 2017). The recurring concept drift framework, RCD, proposed by Gonçalves Jr and De Barros (2013) also uses an explicit approach to handle concept drift. This method extends a similar approach proposed by Widmer and Kubat (1996) that was originally designed to deal with categorical data, to work with numerical data. It stores the context of each data type in a buffer and a classifier is built together with a concept drift detector. This method uses a two-step drift detection mechanism. The drift detector uses a new classifier to detect concept drift. If there is no concept drift, an empty buffer is filled with the samples used in the training of the classifier. Furthermore, there is a warning level to monitor the error rate of the classifier where if the error rate of the classifier reaches the warning level, it means concept drift might be occurring. Simultaneously, a new classifier is created with the new buffer. If the error rate of this classifier decreases and returns to the normal level, then the algorithm will treat that concept drift as a false alarm. Furthermore, when a true concept drift occurs, a statistical test compares the current buffer with all previous buffers to detect whether the new concept has already been observed in the past. If the test result

is positive, which means the new concept has been seen before, then the previous classifier and the buffer corresponding to this concept are used. Otherwise, the current classifier and the buffer are stored in the list.

The main limitation of these explicit methods is that they frequently falsely detect concept drift. Therefore, using these explicit methods may result in incorrect concept drift detection while reducing the model performance. Furthermore, it is difficult to use a drift detector to recognise different concept drift types. Thus, implicit methods are usually more useful in handling concept drift.

The DWM method proposed by Kolter and Maloof (2007) is an implicit method of handling concept drift. This method uses a four-step process to handle concept drift: (1) the DWM holds a weighted pool of base learners which adds or removes base learners based on the performance of the global algorithm; (2) if the global algorithm makes a mistake, then an expert will be added; (3) if a base learner makes a mistake, then its weight will be reduced; and (4) if the weight of a base learner reduces below a threshold, then it will be removed. The SAE method proposed by Gomes et al. (2013) is also a popular implicit method which is also known as the social network abstraction for data stream classification. It uses a similar learning strategy as the DWM method. The SAE method explores the similarities between its classifiers. It also drops classifiers based on their accuracy and similarities with other classifiers. The new classifiers are only added if the overall method accuracy is lower than a given threshold. Furthermore, SAE connects two classifiers when they have very similar predictions.

Even though implicit methods overcome the issues of explicit methods such as the model sensitivity to falsely detecting concept drift and difficulties in recognising different concept drift types, they take a considerable amount of time to adapt to a new concept. This phenomenon has motivated researchers to explore robust and rapidly reactive concept drift handling methods without drift detection.

The EIA method proposed by Baier et al. (2020) is an implicit method that does not use drift detection. The EIA uses two models with different complexities,  $M_{\text{simple}}$  and  $M_{\text{complex}}$ , to make predictions. At a time, the predictions are obtained using a single model where this model is selected based on its error predicted using a regression method. Here,  $M_{\text{complex}}$  captures more information during model training and thus, in normal situations,  $M_{\text{complex}}$  is used to make predictions. However, when the data has concept drift, in particular sudden concept drift,  $M_{\text{complex}}$  is

unable to respond to the concept drift quickly. Thus, in such situations,  $M_{\text{simple}}$  which is trained with most recent observations is applied as it can quickly adapt to the current concept. However, the EIA method is only applicable to sudden concept drift as it is difficult to switch between the two models more frequently to handle continuous drift types such as incremental and gradual concept drift types.

In line with that, in this study, we propose two methods which use the continuous adaptive weighting approach, ECW and GDW, where both methods are inspired by the EIA method. Similar to the EIA, our proposed methods use two sub-models. However, unlike the EIA method, the weighted average of the forecasts provided by both sub-models is considered during forecasting where the weights of the sub-models are dynamically changed. Furthermore, all the above-explained concept drift handling work is from the classification domain where in this study, we explore the elements of these methods that should be changed to adapt them to handle concept drift in the forecasting domain.

## METHODOLOGY

This section explains the series weighted methods and our proposed concept drift handling methods in detail.

### *Series Weighted Methods*

In this simple and straightforward first approach, we assign different weights for the training instances. In particular, we consider the most recent instances to be more relevant than older instances and thus, during training, we assign higher weights for the most recent instances where the weights are gradually decreased for the older instances, e.g. using an exponential decay similar to ETS.

In our work, we consider two series weighting methods: exponential weighting and linear weighting. Equations 7.1 and 7.2, respectively, define the process of weighting the training instances and the method of calculating the weights with exponential and linear weighting methods. Here,  $x_i$  is the original  $i^{\text{th}}$  instance,  $X_i$  is the weighted  $i^{\text{th}}$  instance, `series_length` is the number of training instances considered from a

particular series,  $\alpha_{\text{series\_length}-i}$  is the weight of the  $i^{\text{th}}$  instance,  $\alpha_0$  is the initial weight of the most recent instance, where  $0 < \alpha \leq 1$ , and  $\beta$  is the weight decrease of the linear weighting, where  $0 < \beta \leq 1$ .

$$X_i = \alpha_{\text{series\_length}-i} \cdot x_i \quad (7.1)$$

$$\alpha_i = \begin{cases} \alpha_{i-1} \cdot \alpha_0 & \text{exponential weighting} \\ \alpha_{i-1} - \frac{\beta}{\text{series\_length}} & \text{linear weighting} \end{cases} \quad (7.2)$$

The experiments are conducted considering two values for `series_length`, 200 instances and all instances. The values of  $\alpha_0$  and  $\beta$  are fixed to 0.9 based on our preliminary experiments. Finally, four models are trained with each of our experimental datasets (Section “[Datasets](#)”), considering the two weighting methods: exponential and linear weighting, and two `series_length` values.

### *Our Proposed Concept Drift Handling Methods*

We propose two continuous adaptive weighting methods: ECW and GDW. In particular, our methods train two forecasting models with different series where one model is trained with the recent series observations ( $M_{\text{partial}}$ ) and the other model is trained with all series observations ( $M_{\text{all}}$ ). The weighted average of the predictions provided by  $M_{\text{partial}}$  and  $M_{\text{all}}$  is considered as the final prediction of a given time point where the weights of these models are dynamically changed and calculated based on the previous prediction error. As  $M_{\text{partial}}$  is trained with the recent data, its prediction accuracy may be less than the prediction accuracy of  $M_{\text{all}}$ , however, if the series encounters concept drift, then  $M_{\text{partial}}$  can quickly adapt to the new concept compared to  $M_{\text{all}}$ . In general, the weight of  $M_{\text{partial}}$  is less than the weight of  $M_{\text{all}}$  due to its lower prediction accuracy. However, when a concept drift occurs, our methods rapidly increase the weight of  $M_{\text{partial}}$  which can efficiently remedy the issues of other implicit concept drift handling methods.

We consider Residual Sum of Squares (RSS) as the loss function of  $M_{\text{partial}}$  and  $M_{\text{all}}$ . Equation 7.3 defines RSS where  $y_i$  are the actual values,  $\hat{y}_i$  are the predictions and  $n$  is the number of predictions.

$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (7.3)$$

As we only consider the error of the last prediction for the weight calculations corresponding with the next prediction, Equation 7.3 can be reduced to:

$$RSS_i = (y_i - \hat{y}_i)^2 \quad (7.4)$$

Based on preliminary experiments and to manage complexity of the experiments, we consider 200 time series observations when training the  $M_{\text{partial}}$  model.

Our proposed ECW and GDW methods are different based on the algorithms they use to calculate the weights of  $M_{\text{partial}}$  and  $M_{\text{all}}$ , which we explain in detail in the next sections.

#### *Error Contribution Weighting*

The ECW method directly outputs the prediction of  $M_{\text{all}}$  as the first prediction. From the second prediction onwards, it determines the weights that should be used to combine the predictions of the two sub-models as follows.

The prediction errors of  $M_{\text{partial}}$  and  $M_{\text{all}}$  for the  $i^{\text{th}}$  prediction,  $\epsilon_{p_i}$  and  $\epsilon_{a_i}$ , are defined as shown in Eqs. 7.5 and 7.6, respectively, considering RSS as the loss function.

$$\epsilon_{p_i} = (y_i - \hat{y}_{\text{partial},i})^2 \quad (7.5)$$

$$\epsilon_{a_i} = (y_i - \hat{y}_{\text{all},i})^2 \quad (7.6)$$

The error percentages of  $M_{\text{partial}}$  and  $M_{\text{all}}$  for the  $i^{\text{th}}$  prediction,  $E_{p_i}$  and  $E_{a_i}$ , are defined as shown in Equations 7.7 and 7.8, respectively.

$$E_{p_i} = \frac{\epsilon_{p_i}}{\epsilon_{p_i} + \epsilon_{a_i}} \quad (7.7)$$

$$E_{a_i} = \frac{\epsilon_{a_i}}{\epsilon_{p_i} + \epsilon_{a_i}} \quad (7.8)$$

The weights corresponding with  $M_{\text{partial}}$  and  $M_{\text{all}}$  for the  $i^{\text{th}}$  prediction,  $w_{p_i}$  and  $w_{a_i}$ , can be then defined as shown in Equations 7.9 and 7.10, respectively. In this way, the higher weight is assigned to the model with the lowest previous prediction error.

$$w_{p_i} = E_{a_{i-1}} \quad (7.9)$$

$$w_{a_i} = E_{p_{i-1}} \quad (7.10)$$

Equation 7.11 shows the formula used to obtain the final prediction corresponding with the  $i^{\text{th}}$  time point,  $\hat{y}_i$ , after weighting.

$$\hat{y}_i = w_{p_i} \cdot \hat{y}_{\text{partial}_i} + w_{a_i} \cdot \hat{y}_{\text{all}_i} \quad (7.11)$$

Algorithm 1 shows the process of the ECW method that is used to obtain the prediction for a given time point.

---

**Algorithm 1** Error Contribution Weighting

---

```

1: procedure ECW( $y_{i-1}$ ,  $\hat{y}_{\text{partial}_{i-1}}$ ,  $\hat{y}_{\text{all}_{i-1}}$ ,  $\hat{y}_{\text{partial}_i}$ ,  $\hat{y}_{\text{all}_i}$ ,  $i$ )
2:   if  $i$  is 1 then
3:      $y_i \leftarrow \hat{y}_{\text{all}_i}$ 
4:   else
5:      $\epsilon_{a_{i-1}} \leftarrow \text{calculate\_rss}(y_{i-1}, \hat{y}_{\text{all}_{i-1}})$ 
6:      $\epsilon_{p_{i-1}} \leftarrow \text{calculate\_rss}(y_{i-1}, \hat{y}_{\text{partial}_{i-1}})$ 
7:      $w_{a_i} \leftarrow \frac{\epsilon_{p_{i-1}}}{\epsilon_{p_{i-1}} + \epsilon_{a_{i-1}}}$ 
8:      $w_{p_i} \leftarrow \frac{\epsilon_{a_{i-1}}}{\epsilon_{p_{i-1}} + \epsilon_{a_{i-1}}}$ 
9:      $y_i \leftarrow w_{p_i} \cdot \hat{y}_{\text{partial}_i} + w_{a_i} \cdot \hat{y}_{\text{all}_i}$ 
10:    end if
11:    return  $y_i$ 
12: end procedure
```

---

### *Gradient Descent Weighting*

The GDW method uses gradient descent to determine the weights of the sub-models. The sub-models are initialised with weights of 0.5. This method also directly outputs the prediction of  $M_{\text{all}}$  as the first prediction. From the second prediction onwards, it determines the weights that should be used to combine the predictions of the two sub-models using gradient descent as follows.

The final error of the  $i^{\text{th}}$  prediction,  $\epsilon_i$ , is defined as shown in Equation 7.12 considering RSS as the loss function.

$$\epsilon_i = (y_i - w_{p_i} \cdot \hat{y}_{\text{partial}_i} - w_{a_i} \cdot \hat{y}_{\text{all}_i})^2 \quad (7.12)$$

The gradients calculated from  $M_{\text{partial}}$  and  $M_{\text{all}}$  for the  $i^{\text{th}}$  prediction,  $g_{p_i}$  and  $g_{a_i}$ , are defined as shown in Eqs. 7.13 and 7.14, respectively.

$$g_{p_i} = \nabla \epsilon_i(w_{p_i}) = -2 \cdot \hat{y}_{\text{partial}_i} \cdot \epsilon_i \quad (7.13)$$

$$g_{a_i} = \nabla \epsilon_i(w_{a_i}) = -2 \cdot \hat{y}_{all_i} \cdot \epsilon_i \quad (7.14)$$

To avoid large gaps between two adjacent weights, the gradients are multiplied by an adjusted rate  $\eta$ . The updated weights of  $M_{partial}$  and  $M_{all}$  for the  $i^{th}$  prediction,  $w_{p_i}$  and  $w_{a_i}$ , are then defined as shown in Eqs. 7.15 and 7.16, respectively. Based on our preliminary experiments, the value of  $\eta$  is fixed to 0.01.

$$w_{p_i} = w_{p_{i-1}} - g_{p_{i-1}} \cdot \eta \quad (7.15)$$

$$w_{a_i} = w_{a_{i-1}} - g_{a_{i-1}} \cdot \eta \quad (7.16)$$

Equation 17 shows the formula used to obtain the final prediction corresponding with the  $i^{th}$  time point after weighting.

$$\hat{y}_i = w_{p_i} \cdot \hat{y}_{partial_i} + w_{a_i} \cdot \hat{y}_{all_i} \quad (7.17)$$

Algorithm 2 shows the process of the GDW method that is used to obtain the prediction for a given time point.

---

**Algorithm 2** Gradient Descent Weighting

---

```

1: procedure GDW( $y_{i-1}, \hat{y}_{i-1}, \hat{y}_{partial_{i-1}}, \hat{y}_{all_{i-1}}, \hat{y}_{partial_i}, \hat{y}_{all_i}, w_{p_{i-1}}, w_{a_{i-1}}, \eta, i$ )
2:   if  $i$  is 1 then
3:      $y_i \leftarrow \hat{y}_{all_i}$ 
4:   else
5:      $\epsilon_{i-1} \leftarrow calculate\_rss(y_{i-1}, \hat{y}_{i-1})$ 
6:      $g_{p_{i-1}} \leftarrow -2 \cdot \hat{y}_{partial_{i-1}} \cdot \epsilon_{i-1}$ 
7:      $g_{a_{i-1}} \leftarrow -2 \cdot \hat{y}_{all_{i-1}} \cdot \epsilon_{i-1}$ 
8:      $w_{p_i} = w_{p_{i-1}} - g_{p_{i-1}} \cdot \eta$ 
9:      $w_{a_i} = w_{a_{i-1}} - g_{a_{i-1}} \cdot \eta$ 
10:     $y_i \leftarrow w_{p_i} \cdot \hat{y}_{partial_i} + w_{a_i} \cdot \hat{y}_{all_i}$ 
11:   end if
12:   return  $y_i$ 
13: end procedure

```

---

When training  $M_{partial}$  and  $M_{all}$  models, we consider both exponential and linear weighting methods which internally weight the training instances as explained in section “[Series Weighted Methods](#)”. Thus, the forecasts are obtained from four model combinations, considering two models and two weighting methods. The final forecasts of the ECW and

GDW methods are obtained by averaging the forecasts provided by the above four model combinations.

Our proposed methods are applicable to any GFM including neural networks, machine learning and deep learning models. However, for our experiments, we consider LightGBM (Ke et al., 2017) as the base learner due to its recent competitive performance over the state-of-the-art forecasting models including deep learning models (Januschowski et al., 2021).

## EXPERIMENTAL FRAMEWORK

In this section, we discuss the experimental datasets, error metrics, evaluation method and benchmarks used in our experiments.

### *Datasets*

We are unaware of any publicly available real-world datasets which contain series showing sudden, incremental and gradual concept drift types, adequately, to evaluate the performance of our proposed methods, ECW and GDW. Thus, in this study, we limit ourselves to the use of simulated datasets that contain series representing sudden, incremental and gradual concept drift types.

Let  $ts_1$  and  $ts_2$  be two time series simulated from the same AR(3) data generation process with different initial values and a random seed. The two series are then combined based on the required concept drift type as shown in Eq. 7.18, where  $x_i$  is the  $i^{\text{th}}$  element of the combined series and `series_length` is the length of the series.

$$\begin{aligned} x_i(\text{sudden}) &= \begin{cases} ts_{1i} & \text{if } i < t_{\text{drift}} \\ ts_{2i} & \text{if } i \geq t_{\text{drift}} \end{cases} \\ x_i(\text{incremental}) &= \begin{cases} ts_{1i} & \text{if } i < t_{\text{start}} \\ (1-w)ts_{1i} + (w)ts_{2i} & \text{if } t_{\text{start}} \leq i \leq t_{\text{end}} \\ ts_{2i} & \text{if } i > t_{\text{end}} \end{cases} \\ x_i(\text{gradual}) &= \begin{cases} ts_{1i} & \text{if } I_i = 0 \\ ts_{2i} & \text{if } I_i = 1 \end{cases} \end{aligned} \quad (7.18)$$

Where

$$w = \frac{i - t_{\text{start}}}{t_{\text{start}} - t_{\text{end}}}$$

$I_i$  is random based on probability  $p_i$

$$p_i = \frac{i}{\text{series\_length}}$$

For sudden concept drift, the drift point,  $t_{\text{drift}}$ , is randomly selected. The combined series contains  $ts_1$  before  $t_{\text{drift}}$  and  $ts_2$  on and after  $t_{\text{drift}}$ . For incremental concept drift, the drift starting point,  $t_{\text{start}}$ , and drift ending point,  $t_{\text{end}}$ , are randomly selected. The combined series contains  $ts_1$  before  $t_{\text{start}}$  and  $ts_2$  after  $t_{\text{end}}$ . A linear weighted combination of  $ts_1$  and  $ts_2$  is considered in between  $t_{\text{start}}$  and  $t_{\text{end}}$  where  $w$  and  $(1 - w)$  are the weights of  $ts_2$  and  $ts_1$ , respectively. For gradual concept drift, the combined series contains values from either  $ts_1$  or  $ts_2$  depending on the corresponding  $I_i$  which is randomised based on an increasing probability,  $p_i$ . In general,  $p_i$  is used to represent the probability that an instance belongs to  $ts_2$ . Thus, when  $p_i = 1$  the series ultimately enters the new distribution. Finally, three simulated datasets are created that respectively contain series representing sudden, incremental and gradual concept drift types. Each dataset has 2000 series. The length of each series in all datasets is 2000. The first 1650 data points in each series are used for model training whereas the last 350 data points in each series are reserved for testing. Note that the drift can occur in either training or testing parts of the series.

### Error Metrics

We measure the performance of our models using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE, Sammut and Webb, 2010) which are commonly used error metrics in the time series forecasting field. Equations 7.19 and 7.20, respectively, define the RMSE and MAE error metrics. Here,  $F_k$  are the forecasts,  $Y_k$  are the actual values for the required forecast horizon and  $h$  is the forecast horizon.

$$RMSE = \sqrt{\frac{\sum_{k=1}^h |F_k - Y_k|^2}{h}} \quad (7.19)$$

$$MAE = \frac{\sum_{k=1}^h |F_k - Y_k|}{h} \quad (7.20)$$

Since all these error measures are defined for each time series, we calculate the mean and median values of them across a dataset to measure the model performance. Therefore, four error metrics are used to evaluate each model: mean RMSE, median RMSE, mean MAE and median MAE.

### *Evaluation*

We use prequential evaluation (Dawid, 1984) to evaluate the performance of the models. This procedure is also a common evaluation technique used in the forecasting space, where it is normally called time series cross-validation (Hyndman and Athanasopoulos, 2018). We conduct the prequential evaluation with increasing block sizes (Cerqueira et al., 2020) as follows.

First, the forecast horizon of each series which consists of 350 data points is divided into 7 test sets of 50 data points each. The models are then trained with the training set and 50 predictions are obtained corresponding with the first test set, one at a time iteratively. This procedure is also known as rolling origin without recalibration (Tashman, 2000) as the models are not retrained when obtaining individual predictions. After obtaining the predictions corresponding with the first test set, the actuals corresponding with the first test set are also added to the training set. The models are retrained with the new training set and the 50 predictions corresponding with the second test set are obtained, one at a time iteratively, in a rolling origin with recalibration fashion. This process is repeated until all 350 predictions are obtained corresponding with the complete forecast horizon.

### *Benchmarks*

We consider three types of LightGBM models: plain LightGBM that does not handle concept drift (Plain), LightGBM trained with exponential weighting (EXP) and LightGBM trained with linear weighting (Linear), as the main benchmarks of our study. We also consider three statistical models: AR(3) model, AR(5) model and ETS (Hyndman et al., 2008) as the benchmarks of our study. All statistical models are implemented using the Python package, StatsForecast (Garza et al., 2022).

Each baseline is separately trained considering all data points and the last 200 data points in the training series. Thus, twelve models are finally

considered as benchmarks, considering six LightGBM models and six statistical models.

### *Statistical Testing of the Results*

The non-parametric Friedman rank-sum test is used to assess the statistical significance of the results provided by different forecasting models across time series considering a significance level of  $\alpha = 0.05$ . Based on the corresponding RMSE errors, the methods are ranked on every series of a given experimental dataset. The best method according to the average rank is chosen as the control method. To further characterise the statistical differences, Hochberg's post hoc procedure (García et al., 2010) is used.

## RESULTS AND DISCUSSION

This section provides a comprehensive analysis of the results of all considered models in terms of main accuracy and statistical significance, and later also gives more insights into the modelling.

### *Main Accuracy Results*

Table 7.1 shows the results of all models across sudden, incremental and gradual concept drift types for mean RMSE, median RMSE, mean MAE and median MAE.

The models in Table 7.1 are grouped based on the sub-experiments. The results of the best performing models in each group are italicised, and the overall best performing models across the datasets are highlighted in boldface. The first group contains the statistical baselines we considered which are AR(3), AR(5) and ETS. The second group contains the LightGBM baselines we considered which are the plain LightGBM model (Plain), LightGBM trained with exponential weighting (EXP) and LightGBM trained with linear weighting (Linear). The baseline models also indicate the number of data points used for model training, which is either last 200 data points or all data points. The third group contains our proposed concept drift handling methods, ECW and GDW, which use two LightGBM models to provide forecasts. The hyperparameters of all LightGBM models such as maximum depth, number of leaves, minimum number of instances in a leaf, learning rate and sub-feature leaf

**Table 7.1** Results across sudden, incremental and gradual concept drift types. The best performing models in each group are italicised and the overall best performing models are highlighted in boldface

		<i>Mean RMSE</i>	<i>Median RMSE</i>	<i>Mean MAE</i>	<i>Median MAE</i>
<b>Sudden</b>					
	AR3_200	0.7318	0.5154	0.6184	0.4933
	AR3_All	0.5973	0.5303	<i>0.4724</i>	0.4310
	AR5_200	0.6964	0.6731	0.5723	0.5533
	AR5_All	<i>0.5516</i>	<i>0.5110</i>	0.4736	0.4454
	ETS_200	0.5525	0.5286	0.5025	0.4847
	ETS_All	0.6118	0.5305	0.4883	<i>0.4307</i>
<b>Incremental</b>					
	EXP_200	0.8409	0.7221	0.6830	0.5844
	EXP_All	<i>0.7379</i>	<i>0.5490</i>	<i>0.5939</i>	<i>0.4441</i>
	Linear_200	0.7631	0.6268	0.6137	0.5030
	Linear_All	0.8113	0.6518	0.6431	0.5102
	Plain_200	0.7838	0.6402	0.6273	0.5109
	Plain_All	0.9242	0.7768	0.7334	0.6113
<b>Gradual</b>					
	GDW	<b>0.4056</b>	<b>0.2899</b>	<b>0.3044</b>	<b>0.2288</b>
	ECW	0.5812	0.4162	0.4598	0.3246
	AR3_200	0.5796	0.5428	0.5001	0.4538
	AR3_All	<i>0.5602</i>	<i>0.5266</i>	<i>0.4501</i>	<i>0.4281</i>
	AR5_200	0.6124	0.5977	0.5377	0.5142
	AR5_All	0.5637	0.5472	0.4551	0.4453
	ETS_200	0.5874	0.5536	0.5544	0.5368
	ETS_All	0.5746	0.5424	0.4634	0.4400
	EXP_200	0.7524	0.7183	0.6101	0.5821
	EXP_All	<i>0.6236</i>	<i>0.5548</i>	<i>0.5018</i>	<i>0.4480</i>
	Linear_200	0.6745	0.6245	0.5403	0.5004
	Linear_All	0.7365	0.6569	0.5836	0.5179
	Plain_200	0.6943	0.6392	0.5541	0.5098
	Plain_All	0.8710	0.7848	0.6892	0.6175
	GDW	<b>0.3629</b>	<b>0.3083</b>	<b>0.2805</b>	<b>0.2437</b>
	ECW	0.5541	0.4420	0.4320	0.3444
	AR3_200	0.7931	0.7877	0.7588	0.7480

(continued)

**Table 7.1** (continued)

	<i>Mean RMSE</i>	<i>Median RMSE</i>	<i>Mean MAE</i>	<i>Median MAE</i>
AR3_All	0.7939	0.7903	<b>0.6819</b>	<b>0.6832</b>
AR5_200	0.8011	0.7957	0.7832	0.7795
AR5_All	0.8637	0.8610	0.8531	0.8499
ETS_200	<b>0.7853</b>	<b>0.7810</b>	0.6973	0.6926
ETS_All	0.7936	0.7924	0.7328	0.7297
EXP_200	1.0023	0.9963	0.7810	0.7745
EXP_All	1.0292	1.0260	0.7825	0.7821
Linear_200	<b>0.8998</b>	<b>0.9006</b>	<b>0.6962</b>	<b>0.6961</b>
Linear_All	1.1786	1.1469	0.9170	0.8921
Plain_200	0.9067	0.9058	0.7036	0.7024
Plain_All	1.3042	1.2433	1.0200	0.9722
GDW	<b>0.7168</b>	<b>0.7161</b>	<b>0.4617</b>	<b>0.4605</b>
ECW	0.7716	0.7711	0.5686	0.5674

are tuned using a grid search approach. For our experiments, we use the Python `lightgbm` package.

In the first group, different statistical models show the best performance across different concept drift types. Overall, AR5\_All and AR3\_All, respectively, show the best performance across sudden and incremental concept drift types. Across gradual concept drift, ETS\_200 and AR3\_All, respectively, show the best performance on RMSE and MAE metrics.

In the second group, EXP\_All shows the best performance across sudden and incremental concept drift types whereas Linear\_200 shows the best performance across gradual concept drift. The models that use exponential weighting overall show a better performance than the models that use linear weighting. Across all concept drift types, overall, the methods that use exponential and linear weighting show a better performance than the plain LightGBM models which do not handle concept drift. This shows assigning higher weights for the recent series observations is beneficial in handling concept drift. For different concept drift types, usage of different numbers of training data points shows best results, and this indicates using two models trained with the recent series history and full series history may be a better way to handle concept drift.

In the third group, our proposed GDW method shows the best performance across all concept drift types. In particular, this method outperforms all considered baselines and shows the best performance on

all error metrics. Our proposed ECW method also outperforms all considered baselines on all error metrics across all concept drift types except for 2 cases: AR5\_All and ETS\_200 on mean RMSE across sudden concept drift. This further shows combining the forecasts of two models trained with the most recent series history and full series history is a proper way of handling concept drift in global time series forecasting. The model trained with the full series history has a high forecasting accuracy where the model trained with the recent series history can quickly adapt to new concepts. Thus, combining the forecasts provided by these two models leads to high forecasting accuracy in particular for the series showing concept drift.

The performance improvements gained by the GDW method across different concept drift types are relative to the characteristics of the drift type. In particular, compared to the LightGBM baselines, the GDW method shows considerable improvements across the sudden and incremental concept drift types and smaller improvements across the gradual concept drift type in terms of mean RMSE and mean MAE. The drift occurs gradually in the series showing gradual concept drift, and thus, there is a possibility that the models have seen instances of the new distribution during training. Hence, the baseline LightGBM models have also provided accurate forecasts up to some extent for the series showing gradual concept drift. However, with the series showing sudden or incremental concept drift, there is a high possibility that the models have seen fewer or no instances of the new distribution during training where the drift should be properly handled by the models to provide accurate forecasts. Our proposed GDW method can properly handle concept drift compared to the baselines and thus, across sudden and incremental concept drift types, it shows considerable performance improvements in terms of the forecasting accuracy.

### *Statistical Testing Results*

Table 7.2 shows the results of the statistical testing evaluation, namely the adjusted  $p$ -values calculated from the Friedman test with Hochberg's post hoc procedure considering a significance level of  $\alpha = 0.05$  (García et al., 2010). The statistical testing is separately conducted using the three experimental datasets corresponding with sudden, incremental and gradual concept drift types. For a given dataset, the RMSE values of each series provided by all methods are considered.

**Table 7.2** Results of statistical testing across sudden, incremental and gradual concept drift types

<i>(a) Sudden concept drift</i>		
<i>Method</i>	<i>pHoch</i>	
<i>GDW</i>	—	
ECW	$3.56 \times 10^{-9}$	
EXP_All	$< 10^{-30}$	
Linear_200	$< 10^{-30}$	
Plain_200	$< 10^{-30}$	
Linear_All	$< 10^{-30}$	
EXP_200	$< 10^{-30}$	
Plain_All	$< 10^{-30}$	
AR3_200	$< 10^{-30}$	
AR3_All	$< 10^{-30}$	
AR5_200	$< 10^{-30}$	
AR5_All	$< 10^{-30}$	
ETS_200	$< 10^{-30}$	
ETS_All	$< 10^{-30}$	

<i>(b) Incremental concept drift</i>		
<i>Method</i>	<i>pHoch</i>	
<i>GDW</i>	—	
ECW	$3.48 \times 10^{-20}$	
EXP_All	$< 10^{-30}$	
Linear_200	$< 10^{-30}$	
Plain_200	$< 10^{-30}$	
EXP_200	$< 10^{-30}$	
Linear_All	$< 10^{-30}$	
Plain_All	$< 10^{-30}$	
AR3_200	$< 10^{-30}$	
AR3_All	$< 10^{-30}$	
AR5_200	$< 10^{-30}$	
AR5_All	$< 10^{-30}$	
ETS_200	$< 10^{-30}$	
ETS_All	$< 10^{-30}$	

<i>(c) Gradual concept drift</i>		
<i>Method</i>	<i>pHoch</i>	
<i>GDW</i>	—	
ECW	$8.01 \times 10^{-8}$	

(continued)

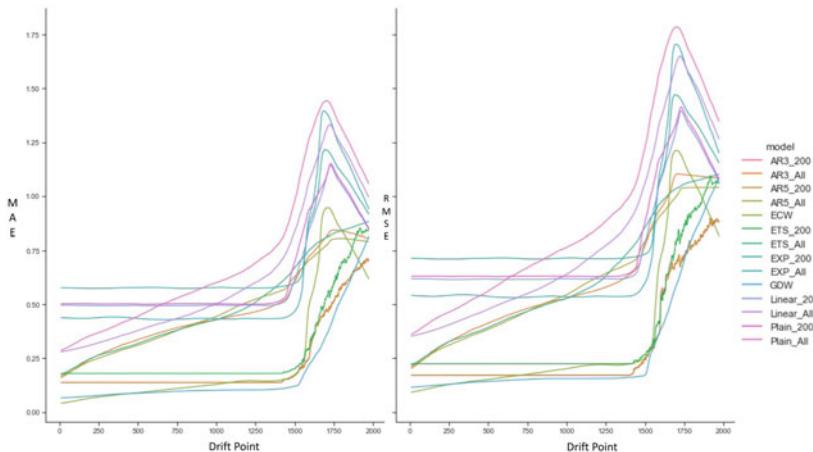
**Table 7.2** (continued)

(c) <i>Gradual concept drift</i>	
<i>Method</i>	<i>p<sub>Hoch</sub></i>
GDW	—
Linear_200	$< 10^{-30}$
Plain_200	$< 10^{-30}$
EXP_200	$< 10^{-30}$
EXP_All	$< 10^{-30}$
Linear_All	$< 10^{-30}$
Plain_All	$< 10^{-30}$
AR3_200	$< 10^{-30}$
AR3_All	$< 10^{-30}$
AR5_200	$< 10^{-30}$
AR5_All	$< 10^{-30}$
ETS_200	$< 10^{-30}$
ETS_All	$< 10^{-30}$

The overall  $p$ -values of the Friedman rank-sum test corresponding with the sudden, incremental and gradual concept drift types are less than  $10^{-30}$  showing the results are highly significant. For all concept drift types, GDW performs the best on ranking over RMSE per each series, and thus, it is used as the control method as mentioned in the first row in each sub-table of Table 7.2. All benchmarks and ECW show  $p_{Hoch}$  values less than  $\alpha$  across all concept drift types, and thus, they are significantly worse than the control method.

### *Further Insights*

Figure 7.2 shows the performance of all models in terms of MAE and RMSE, across the series showing sudden concept drift with different drift points. According to the figure, our proposed methods, GDW and ECW, show a better performance compared to the baselines in both RMSE and MAE where the GDW shows the best performance in both error metrics. Compared to the ECW and other baselines, the GDW shows a lower error increasing rate. All methods show higher errors for the series where the concept drift occurs in the test set (after 1650 data points) as the models have not seen the corresponding data distributions in the training



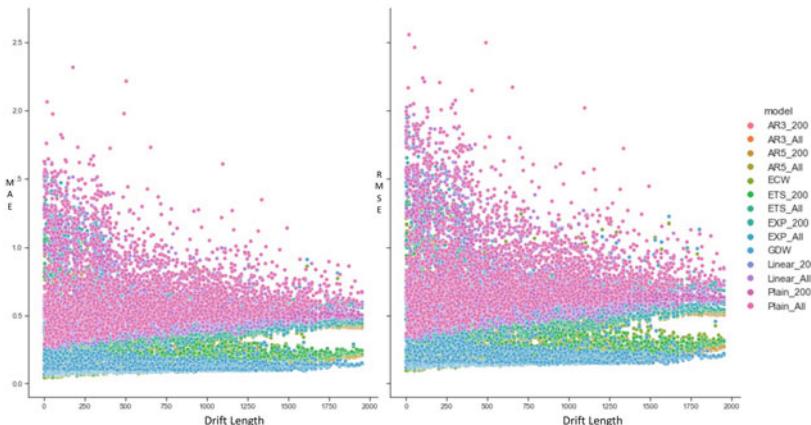
**Fig. 7.2** Performance of all models in terms of MAE (left) and RMSE (right), across the series showing sudden concept drift with different drift points

set. However, even in this phenomenon, we see that the error increasing rate of the GDW method is considerably lower compared with the other methods.

Figure 7.3 shows the performance of all models in terms of MAE and RMSE, across the series showing incremental concept drift with different drift lengths. Here, the drift length refers to the difference between the drift starting and ending points. According to the figure, we see that all models show higher forecasting errors for shorter drift lengths, as the models are required to adapt to the new concept as quickly as possible for shorter drift lengths. Our proposed methods, GDW and ECW, are continuous adaptive weighting methods, and they continuously adjust model weights based on the previous prediction error. Thus, GDW and ECW show a better performance compared to the other methods where GDW shows the best performance in both MAE and RMSE.

## CONCLUSIONS AND FUTURE RESEARCH

Time series data distributions are often not stationary, and as a result, the accuracy of ML forecasting models can decrease over time. Thus, handling such concept drift is a crucial issue in the field of ML methods



**Fig. 7.3** Performance of all models in terms of MAE (left) and RMSE (right), across the series showing incremental concept drift with different drift lengths

for time series forecasting. However, the concept drift handling methods in the literature are mostly designed for classification tasks, not for forecasting.

In this study, we have proposed two new methods based on continuous adapting weighting, GDW and ECW, to handle concept drift in global time series forecasting. Each proposed method uses two sub-models, a model trained with the most recent series history and a model trained with the full series history, where the weighted average of the forecasts provided by the two models is considered as the final forecasts. The proposed methods calculate the weights that are used to combine the sub-model forecasts based on the previous prediction error. During training, the sub-models internally assign higher weights for the recent observations either exponentially or linearly. Across three simulated datasets that demonstrate sudden, incremental and gradual concept drift types, we have shown that our proposed methods can significantly outperform a set of baseline models. In particular, GDW shows the best performance across all considered concept drift types.

From our experiments, we conclude that using two models trained with training sets of different sizes is a proper approach to handle concept drift in global time series forecasting. We further conclude gradient descent is a proper approach to determine the weights that should be used

to combine the forecasts provided by these models. Thus, we recommend the GDW method to handle concept drift in global time series forecasting.

The success of this approach encourages as future work to explore the performance of GDW and ECW methods when using three or more sub-models trained using different numbers of data points. Analysing the performance of the proposed methods based on the characteristics of the datasets such as the number of series, and the order and parameters of the AR data generation process are also worthwhile to study. It will be also interesting to study the performance of our proposed methods on real-world datasets that adequately demonstrate concept drift. Analysing the performance of our proposed methods across different base learners including deep learning models is also a worthwhile endeavour.

## REFERENCES

- Baier, L., Hofmann, M., Kühl, N., Mohr, M., & Satzger, G. (2020). Handling concept drifts in regression problems—the error intersection approach. In *15th International Conference on Wirtschaftsinformatik*.
- Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140, 112896.
- Canseco, M. M., & Garza, F. (2022). *Nixtla: Transfer learning for time series forecasting*. <https://github.com/Nixtla/transfer-learning-time-series>
- Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11), 1997–2028.
- Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Canseco, M. M., & Dubrawski, A. (2023). N-HiTS: Neural hierarchical interpolation for time series forecasting. In *AAAI Conference on Artificial Intelligence* (Vol. 37).
- Chu, F., & Zaniolo, C. (2004). Fast and light boosting for adaptive mining of data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 282–292). Springer.
- Dawid, A. P. (1984). Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2), 278–290.
- Delany, S. J., Cunningham, P., Tsymbal, A., & Coyle, L. (2004). A case-based technique for tracking concept drift in spam filtering. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence* (pp. 3–16). Springer.
- García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in

- computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10), 2044–2064.
- Garza, F., Canseco, M. M., Challú, C., & Olivares, K. G. (2022). *StatsForecast: Lightning fast forecasting with statistical and econometric models*. PyCon Salt Lake City, Utah, US.<https://github.com/Nixtla/statsforecast>
- Ghomeshi, H., Gaber, M. M., & Kovalchuk, Y. (2019). EACD: Evolutionary adaptation to concept drifts in data streams. *Data Mining and Knowledge Discovery*, 33(3), 663–694.
- Godahewa, R., Bandara, K., Webb, G. I., Smyl, S., & Bergmeir, C. (2021). Ensembles of localised models for time series forecasting. *Knowledge-Based Systems*, 233, 107518.
- Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2), 1–36.
- Gomes, H. M., & Enembreck, F. (2013). SAE: Social adaptive ensemble classifier for data streams. In *IEEE Symposium on Computational Intelligence and Data Mining* (pp. 199–206).
- Gonçalves Jr, P. M., & De Barros, R. S. S. (2013). RCD: A recurring concept drift framework. *Pattern Recognition Letters*, 34(9), 1018–1025.
- Grazzi, R., Flunkert, V., Salinas, D., Januschowski, T., Seeger, M., & Archambault, C. (2021). *Meta-forecasting by combining global deep representations with local adaptation*. <https://arxiv.org/abs/2111.03418>
- Hewamalage, H., Bergmeir, C., & Bandara, K. (2020). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*.
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts.
- Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: The state space approach*. Springer Science and Business Media.
- Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1), 167–177.
- Januschowski, T., Wang, Y., Torkkola, K., Erkkilä, T., Hasson, H., & Gasthaus, J. (2021). Forecasting with trees. *International Journal of Forecasting*.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17 (pp. 3149–3157). Curran Associates Inc.
- Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8, 2755–2790.

- Krawczyk, B., Minku, L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37, 132–156.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4), 802–808.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). The M5 accuracy competition: Results, findings and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364.
- Mean Absolute Error. (2010). *Encyclopedia of machine learning* (C. Sammut & G. I. Webb, Eds., p. 652). Springer.
- Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*. ISSN 0169-2070.
- Oreshkin, B. N., Carpow, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations (ICLR)*.
- Oreshkin, B. N., Carpow, D., Chapados, N., & Bengio, Y. (2021). Meta-learning framework with applications to zero-shot time-series forecasting. In *AAAI Conference on Artificial Intelligence* (Vol. 35).
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.
- Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85.
- Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, 16(4), 437–450.
- Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4), 964–994.
- Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1), 69–101.
- Woo, G., Liu, C., Sahoo, D., Kumar, A., & Hoi, S. (2022). *Deeptime: Deep time-index meta-learning for non-stationary time-series forecasting*. <https://arxiv.org/abs/2207.06046>
- Ye, R., & Dai, Q. (2018). A novel transfer learning framework for time series forecasting. *Knowledge-Based Systems*, 156, 74–99.



## CHAPTER 8

---

# Neural Network Ensembles for Univariate Time Series Forecasting

*Artemios-Anargyros Semenoglou, Evangelos Spiliotis,  
and Vassilios Assimakopoulos*

## INTRODUCTION

Time series forecasting has been adopted as a critical tool for better decision-making in many different domains. Naturally, this increase in popularity has led more researchers to focus on expanding the toolkit of available methods by developing new, more accurate forecasting methods. As a result, practitioners now have a large number of methods to choose from, each one more suitable for a particular task. At the same time,

---

A.-A. Semenoglou (✉) · E. Spiliotis · V. Assimakopoulos  
Forecasting and Strategy Unit, School of Electrical and Computer Engineering,  
National Technical University of Athens, Athens, Greece  
e-mail: [artemis@fsu.gr](mailto:artemis@fsu.gr)

E. Spiliotis  
e-mail: [spiliotis@fsu.gr](mailto:spiliotis@fsu.gr)

V. Assimakopoulos  
e-mail: [vassim@fsu.gr](mailto:vassim@fsu.gr)

as new models are being developed, performance improvements can be realized by employing alternative strategies, such as the combination of forecasts generated by multiple, different forecasting models.

The practice of combining forecasts generated by groups (ensembles) of models is considered to be one the most successful strategies for improving forecasting accuracy and reducing error variability (Wang et al., 2022). The origins of this strategy, in its modern form, can be traced back in the work of Bates and Granger (1969), who concluded that it is possible to combine two sets of forecasts, in order to produce a final set of forecasts that is more accurate than the constituent sets. Since then, a significant number of experimental studies (Lemke & Gabrys, 2010; Gastinger et al., 2021) and forecasting competitions (Makridakis & Hibon, 2000; Makridakis et al., 2018) have provided solid evidence supporting this approach, over alternatives that rely on selecting a single model for forecasting a target time series.

Traditionally, combining different sets of forecasts aims at overcoming the biases of specific forecasting methods and removing the burden of identifying the single “best” one, that is the most accurate when it comes to forecasting the target series (Kourentzes et al., 2019), a problem that is often impossible to solve. In their study, Petropoulos et al. (2018) extensively discuss the mechanisms behind different sources of uncertainty which are present in every time series forecasting task due to randomness. Particularly, the uncertainty related to selecting a forecasting model and determining its parameters may lead to the use of “sub-optimal” models and in forecasts with lower accuracy. They find that these issues can be mitigated by constructing ensembles of different models fitted with different parameters.

Although the core idea of combining different sets of forecasts is quite simple, a large number of combination approaches have been proposed over the years, ranging from straightforward forecast averages to elaborate combination schemes based on machine learning meta-learners (Wang et al., 2022). For example, Petropoulos and Svetunkov (2020) simply averaged the forecasts of four statistical methods, using the median operator, and managed to outperform a large number of submissions in the M4 competition. Similarly, Kourentzes et al. (2014) examined different operators for combining sets of forecasts produced by neural networks, and concluded that using median-based or mode-based combinations results in significant accuracy increases. Aiolfi and Timmermann (2006) proposed a more complex combination scheme that relies on

weighting the forecasts of different models based on their past performance, resulting in more accurate forecasts when compared to simpler combination schemes. Following a different approach, Kang et al. (2022) argued in favor of using the out-of-sample forecasts of the constituent models, in order to calculate combination weights that maximize the diversity of the final set of forecasts. Finally, Montero-Manso et al. (2020), the runner-up of the M4 competition, used a machine learning meta-learner for calculating the optimal weights in order to combine a number of statistical forecasting methods, based on the characteristics of each time series.

More recently, the introduction and widespread adoption of machine learning (ML) and neural network (NN) models expanded the discussion around the need for ensembles of models. In theory, the process of fitting NNs should be more resilient to uncertainties related to model and parameter estimation, as these models are not bounded by hard assumptions about the underlying data generative processes. Essentially, NNs can be trained into the “best” model by recognizing patterns found in the data (Barker, 2020). However, as NNs were introduced, a different set of challenges emerged that need to be addressed in order to further boost the accuracy of their forecasts. The first challenge relates to the stochastic nature of NNs. The element of randomness is at the core of NNs, affecting different aspects such as the initial network parameters as well as the order in which training samples are introduced during learning. As a result, each time the same network architecture is trained using a different seed, the final model and its forecasting performance will be different (Ahmed & Lofstead, 2022). A second challenge concerns the selection of a particular NN configuration by the developer. Although NNs approach each task without making any data-related assumptions, the structure of the problem, as introduced to the models, is determined by their configuration. For example, in auto-regressive forecasting NNs, hyper-parameters such as the size of the input vector and the forecasting horizon transform the underlying problem and the resulting model, by providing a different amount of historical observations and altering the expected outcome, respectively. Similarly, depending on the loss function specified for training, the model is optimized according to the particularities of the specific metric. Drawing from the literature on ensembles of statistical forecasting methods, researchers have concluded that combining sets of forecasts generated by multiple NNs can be an effective strategy for dealing with these NN-specific challenges.

As more forecasting NN architectures become mainstream, relevant research has expanded to include strategies for constructing ensembles of NNs, claiming further performance improvements. Several approaches for creating diverse ensembles of NNs can be found in the literature, with most of them leveraging on the architecture of the models, the parameters of the training process, and the set of data used during training. For example, in their study, Makridakis et al. (2022) considered a deep auto-regressive recurrent neural network (DeepAR), a Transformer model, a WaveNet model, and a simple multi-layer perceptron (MLP) in order to forecast the time series of the M3 competition. They reported significant accuracy improvements when the forecasts produced by these different classes of NNs were combined. Choi and Bumshik (2018) suggested an alternative strategy for creating diverse sets of forecasts. Using only long short-term memory (LSTM) models, they implemented ten variants, with a different number of memory cells for each LSTM model. The generated sets of forecasts were subsequently combined based on the performance of each variant on a validation set of series. Another NN-based ensemble approach was developed by B. Trotta in the M4 competition and ranked as the best “pure” ML entry.<sup>1</sup> The pool of models consisted of NNs that differed as follows: some NNs only consisted of fully connected layers while other NNs featured convolutional layers, the number of past observations that each NN received as input differed, and each NN was trained to predict a particular forecasting horizon. All the generated sets of forecasts were simply averaged at the end. Likewise, N-BEATS, introduced by Oreshkin et al. (2019), was an ensemble of deep NNs. Although all models shared a similar architecture, their exact configurations were different. By considering input windows of 6 different lengths, 3 loss functions for training the networks, 2 functions for calculating each model’s forecasts, and 10 random initializations for each configuration, the ensemble was able to achieve state-of-the-art forecasting performance. A different approach was proposed by Godahewa et al. (2021), as they examined various techniques for clustering the time series of a larger set, effectively creating several smaller training sets, based on the number of clusters requested and the given seeds. Multiple NNs were then trained on different clusters of series and combined in order to produce the final forecasts.

<sup>1</sup> For more information about B. Trotta’s submission, see The M4 Team (2018).

The studies discussed above are only a fraction of the available relevant literature. They are, however, indicative of the widespread adoption of NNs and their ensembles in forecasting tasks. Many state-of-the-art NN forecasting methods are based on novel architectures but they, ultimately, rely on combining several models. Despite that, it is often the case that the discussion around the advantages of a particular NN architecture overshadows the analysis of the benefits from combining multiple models. Motivated by these observations, our aim is to examine in greater detail the combination of sets of forecasts produced by NNs. The contributions of this paper are threefold:

- We investigate the potential improvements in forecasting accuracy and robustness by creating simple and easy-to-implement ensembles of NNs. These ensembles consist of relatively shallow, feed-forward, and fully connected NNs, with lower computational requirements for training and deploying them. We also adopt the simplest combination scheme that uses the median operator for combining multiple sets of forecasts, instead of a more sophisticated scheme.
- We examine different configurations for creating diverse pools of forecasting NNs. Specifically, we focus on changing three hyperparameters: the seed for initializing the models' weights and training, the size of their input layer, and the training loss function.
- We empirically assess the performance of individual NNs and their ensembles. For that purpose, we consider three large, publicly available sets of time series: the yearly, quarterly, and monthly series of the M4 competition (Makridakis et al., 2020). We measure and discuss the forecasting accuracy, the distribution of the errors as well as the computational cost of the tested approaches.

The rest of the paper is organized as follows: Section “[Experimental Design](#)” presents the experimental design that was adopted, in order to measure the effect from employing ensembles of forecasting NNs. Section “[Empirical Evaluation](#)” provides details on the implementation of the experimental design. This includes the three sets of series used for evaluating the performance of individual models and their ensembles, the full architecture and training process for the NNs, and the performance measures utilized. Section “[Results and Discussion](#)” presents and discusses

the results. Finally, section “[Conclusions](#)” summarizes the key conclusions as well as some potential paths for future research.

## EXPERIMENTAL DESIGN

As the relevant literature expands, several classes of NNs have already been introduced and are now available for use in forecasting applications. From shallow feed-forward to deep learning recurrent and convolutional networks, the list of candidate architectures is quite extensive. Furthermore, for each class of NNs, a large number of hyper-parameters enable practitioners to create further variants from a base NN, with significantly different forecasting performances. This means that, in theory, an endless pool of candidate models is available for building an ensemble of NNs. However, in practice, it is impossible to consider every possible NN variant. For the purposes of the present paper, we focus on specific NN aspects for developing a diverse set of forecasting models. In this section, we describe in detail the experimental design that was followed, in order to measure any positive or negative impact from employing ensembles of forecasting NNs.

First, we establish a simple forecasting NN, described in greater detail in section “[Base Forecasting Neural Network](#),” as the baseline model and the point of origin for creating additional NN variants. Then, we consider three options for diversifying the base forecasting NN, namely building and training models with different initial seeds, building models with different input layer sizes, and training models with different loss functions. For each one of these three options, we examine the forecasting performance of individual NN models and their respective ensembles. Finally, we also study a final ensemble that combines all NN variants, created by tinkering with all three options.

In order to generate the final set of forecasts of an ensemble, all participating sets of forecasts are combined using the median operator. The combined forecasts are calculated as follows:

$$\hat{y}_{t,\text{median}} = \text{Median}(\hat{y}_{t,1}, \hat{y}_{t,2}, \dots, \hat{y}_{t,n})$$

where  $\hat{y}_{t,\text{median}}$  is the forecast of the ensemble at point  $t$  and  $\hat{y}_{t,n}$  is the forecast at point  $t$  produced by the  $n$ -th model participating in the ensemble.

### *Initialization Seed*

The simplest approach for deploying multiple forecasting NNs that produce different sets of forecasts is to initialize their topology and training with various random seeds. Given a particular architecture, the provided seed determines the initial values of the network's parameters. This means that each NN instance starts training from a different point in the parameters' space and, therefore, it finishes training at a different point compared to NN instances initialized with different seeds (Narkhede et al., 2022). As a result, since the final parameters of each NN are not the same, the generated sets forecasts will also differ. Furthermore, most modern optimization algorithms are stochastic in nature. They randomly shuffle and split samples in batches that are then introduced to the network under training. Thus, at each training step, all models' parameters will diverge, depending on the composition of each batch. Both these aspects make the expected performance of NNs less predictable, compared to other machine learning algorithms. Our aim is to investigate whether it is possible to reduce the impact of randomness on the accuracy of the final forecasts by embracing the inherent unpredictability of NNs and combining models with various seeds.

In this paper, we develop forecasting NNs that share the same architecture and training, as described in section “[Base Forecasting Neural Network](#)”, but they use different initial seeds. First, we develop 200 randomly initialized base NNs and compare them to 200 ensembles of 10 randomly initialized base NNs each. This direct comparison of identical-sized populations can provide insights on potential improvements in forecasting accuracy, as a consequence of ensembling.

Furthermore, we examine the relation between the size of an ensemble and changes in forecasting accuracy and robustness. In order to do so, we create 20 ensembles consisting of different randomly initialized base NNs. Starting from trivial single-model ensembles, we increase the size of the ensembles by adding more NNs to them, iteratively, until they consist of 200 randomly initialized NNs each. At each step of the process, we can link performance changes to the size of the ensembles, by aggregating the performance of all 20 ensembles.

### *Loss Function*

Modern NNs are trained using optimization algorithms that implement some form of stochastic gradient descent. This means that while designing NNs, practitioners are required to select a function that measures the error at each training step and provides directions for further improving the model (loss function). Since the selected loss function guides the training process, it should be reflective of the objectives set, in the context of a particular problem. For example, in forecasting applications, depending on whether it is more important to minimize the possibility of large forecast errors or to maximize the forecasting performance in the majority of the series, different loss functions may be more appropriate for training. In another time series-related example, the application of scaling techniques, on the one hand, enables the use of certain scale-dependent loss functions, such as MAE and MSE. On the other hand, since the target range for most scalers is close to zero, this often comes at the expense of using loss functions based on percentages. Overall, identifying the most suitable loss function among a large number of candidates can be a challenging task.

When it comes to compiling a list of candidate loss functions to be used in this study, we consider the specific characteristics of the forecasting problem at hand. First, we intend to focus on the performance NNs and their ensembles, while generating point forecasts. Furthermore, taking into account the latest trends in the forecasting literature, our NNs are trained in a cross-learning fashion using samples from multiple series of different scales, in order to be able to produce forecasts for the entire set of series. Finally, the time series used for training and evaluating the NNs are not scaled, normalized, or seasonally adjusted. Thus, the pool of losses that would be reasonable to consider is more limited. We implement NNs using the following loss functions:

- **MAPE:** The mean absolute percentage error (MAPE) is an intuitive and scale-independent accuracy measure that is often used in regression problems. It is defined, as a loss function, as follows:

$$L_{\text{MAPE}} = \frac{1}{n} \sum_{i=1}^n \frac{|Y_i - F_i|}{|Y_i|} \times 100\% \quad (8.1)$$

where  $F_i$  is the  $i$ -th value predicted by the model,  $Y_i$  is the corresponding  $i$ -th expected output, and  $n$  is the number of outputs.

- **sMAPE:** The symmetric mean absolute percentage error (sMAPE) was introduced as an alternative to MAPE (Makridakis, 1993). It is a well-known accuracy measure in forecasting, and it was used in the M3 and the M4 forecasting competitions. It is defined as follows:

$$L_{\text{sMAPE}} = \frac{2}{n} \sum_{i=1}^n \frac{|Y_i - F_i|}{|Y_i| + |F_i|} \times 100\% \quad (8.2)$$

where  $F_i$  is the  $i$ -th value predicted by the model,  $Y_i$  is the corresponding  $i$ -th expected value, and  $n$  is the number of outputs.

- **MASE:** The mean absolute scaled error (MASE) is another accuracy measure that it is often used in forecasting-related studies and was one of the official measures of the M4 competition. It was introduced in an attempt to fix some of the flaws of measures, such as the MAPE and sMAPE (Hyndman & Koehler, 2006). MASE is a time series-specific measure and its definition relies on past observations that are simultaneously provided as inputs to the network. It is defined as follows:

$$L_{\text{MASE}} = \frac{\frac{1}{n} \sum_{i=l+1}^{l+n} |Y_i - F_i|}{\frac{1}{l-m} \sum_{i=m+1}^l |Y_i - Y_{i-m}|}, \quad (8.3)$$

where  $F_i$  is the  $i$ -th value predicted by the model,  $Y_i$  is the corresponding  $i$ -th expected value,  $n$  is the number of outputs,  $l$  the number of historical observations provided as inputs to the network, and  $m$  the frequency of the series included in the data set.

- **scMAE:** The scaled mean absolute error (scMAE) is the last measure considered in this study. Although MAE is popular in regression tasks, since it is not scale-independent, it cannot be used in our experiments. Consequently, we adapted MAE by scaling its value, using the mean level of the expected outputs. It is defined as follows:

$$L_{\text{scMAE}} = \frac{1}{n} \sum_{i=1}^n \frac{|Y_i - F_i|}{\bar{Y}} \quad (8.4)$$

where  $F_i$  is the  $i$ -th value predicted by the model,  $Y_i$  is the corresponding  $i$ -th expected value,  $n$  is the number of outputs, and  $\bar{Y}$  is the mean of the expected output values for the sample.

Concerning the loss functions mentioned above, first, we train 10 base forecasting NNs using one of the four loss functions. Then, we build four loss function-specific ensembles, with each ensemble consisting of 10 NNs trained with a particular loss function. Finally, we build an ensemble consisting of 40 NNs, trained with all four loss functions. This setup allows us to compare the forecasting performance of individual models and their ensembles when different loss functions are used. The final ensemble provides insights on the benefits from combining NNs trained with different loss functions.

### *Input Layer Size*

The last hyper-parameter we investigate in greater detail is that of the size of the networks' input layer. In practice, this hyper-parameter controls the amount of past observations provided to the model as input. A larger input layer means that the corresponding NN generates its forecasts by looking at longer historical windows. One the one hand, additional past observations may reveal more subtle and long-term patterns that can be leveraged for more accurate forecasts. On the other hand, information from the most recent observations may be diluted, as more data points have to be processed by the NN. Furthermore, increasing the number of the inputs, in turn, reduces the amount of training samples that can be extracted from each time series. This is a critical trade-off to consider, especially when dealing with relatively short business time series. Selecting an input size can be challenging as balance is required in order for each sample to contain enough information, without excessively limiting the total number of samples.

For our experiments, we developed NNs with seven different input layer sizes. In each experiment, the frequency of the target time series and the requested forecasting horizon determined the exact size of the input vector. The examined input layer sizes ranged, from 1, up to 7 times the forecasting horizon.

Also, similarly to the design described in section “[Loss Function](#),” we develop 10 base forecasting NNs for each of the seven input layer sizes considered. Subsequently, we combine the forecasts of NNs that have the

same input layer size, in order to create one ensemble of models per size. This particular design allows us to assess the forecasting performance of individual models and their ensembles, for different input sizes. Finally, we combine all 70 sets of forecasts, regardless of the size of the networks' input layer, creating a final ensemble consisting of models that leverage different numbers of past observations. The final ensemble allows us to examine the benefits from combining NNs that are exposed to different historical windows.

## EMPIRICAL EVALUATION

In this section, more detailed information is provided on the implementation of the experimental design that is used for assessing the accuracy of the individual forecasting models and the respective ensembles. We present the data sets used for training and evaluating the performance of the tested approaches, the forecasting performance measures utilized, and the architecture of the individual base forecasting NNs.

### *Time Series Data*

In order to evaluate the potential benefits from deploying ensembles of NNs, we consider the time series of the M4 competition (Makridakis et al., 2020). This data set consists of a large number of time series from different domains (finance, micro, macro, industry, and demographics), and, as a result, it allows us to safely generalize the findings of our work. Moreover, since the data set is publicly available and widely utilized in time series forecasting studies, our experiments can be easily reproduced and the results can be compared with those from other relevant studies.

We use the 23,000 yearly, the 24,000 quarterly, and the 48,000 monthly series of the M4 competition, to be noted *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly*, respectively. For all three subsets, we follow the original design of the M4 competition. This means that the forecasting horizon,  $h$ , is set to 6 years, 8 quarters, and 18 months for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets, respectively. The last  $h$  observations of the series are considered out-of-sample observations and they are used solely for evaluating the performance of the examined forecasting approaches. The rest of the time series observations (in-sample) are used for optimizing and training the forecasting models.

For constructing the respective training data sets, when a sufficient number of data points is available, a sliding window strategy (Smyl, 2020) is employed on the in-sample part of each time series, in order to increase the total number of training samples. Note that the length of the sliding window varies across the different experiments, as it is dependent on the frequency of the series and the input layer size of the NNs, as described in section “[Input Layer Size](#).” This means that for the series in *M4\_Yearly*, with a forecasting horizon of 6 years, we consider windows of 6, 12, 18, 24, 30, 36, and 42 observations. For the series in *M4\_Quarterly*, with a forecasting horizon of 8 quarters, the windows contain 8, 16, 24, 32, 40, 48, or 56 data points. For the series in *M4\_Monthly*, with a forecasting horizon of 18 months, each sample contains 18, 36, 54, 72, 90, 108, or 126 observations.

It is also important to highlight that no scaling, normalization, or decomposition techniques are used on the time series data before they are introduced to the NNs. Although the use of pre-processing techniques, such as scaling and seasonal decomposition, is quite common when NNs are involved (Barker, 2020; Zhang & Qi, 2005), more recent studies successfully employ complex NNs trained in cross-learning fashion without any data pre-processing (Li et al., 2022; Oreshkin et al., 2019). As a result, we opted for the latter approach as it is simpler and easier to implement.

### *Forecasting Performance Measures*

To assess the accuracy of the forecasts produced by different NN ensembles, we employ the mean absolute scaled error (MASE) (Hyndman & Koehler, 2006), as it was used in the M3 and M4 forecasting competitions. This measure is often used in time series forecasting literature, making our results comparable with other relevant studies. It is defined as:

$$\text{MASE} = \frac{\frac{1}{h} \sum_{t=n+1}^{n+h} |Y_t - F_t|}{\frac{1}{n-1} \sum_{t=2}^n |Y_t - Y_{t-1}|},$$

where  $F_t$  is the forecast of the examined approach at point  $t$ ,  $Y_t$  is the actual value of the series at point  $t$ ,  $h$  is the forecasting horizon that corresponds to the frequency of the data, and  $n$  is the number of available historical observations of the series. In order to measure the overall

forecasting accuracy, we follow the guidelines of the M4 competition. We first calculate MASE for each individual time series, and, then, we average these errors across the entire collection of series.

While studying the potential benefits from combining multiple models, apart from the accuracy of the forecasts, it is also important to discuss the computational time needed for training and using additional models. In the present study, we report computational time in minutes required for training the NNs involved in an ensemble as well as for generating and combining their forecasts. Note that the reported time corresponds to the time needed for sequentially training all participating NNs and extracting their forecasts. Depending on the available computational resources, the actual time for employing an ensemble can be significantly reduced by deploying models in parallel. All NNs were GPU-trained using a system with the following specifications: 2 Intel Xeon Gold 5118 CPUs @ 2.30GHz, x64 based processor, 24 cores, 48 logical processors, 64.00 GB RAM, and a Quadro RTX 5000 GPU with 16 GB of VRAM.

### *Base Forecasting Neural Network*

The NNs participating in the forecasting ensembles are simple, feed-forward networks, with fully connected layers. Although recently more and more research is focused on developing deep NNs, shallow multi-layer perceptrons can be effectively used instead, especially when the forecasting task at hand involves a large number of relatively short time series. Their forecasting performance is close to that of state-of-the-art deep learning alternatives, while being easier to implement and faster to train. All models used in the ensembles were developed in Python 3.7.6 using Google's Tensorflow API v.2.4.0.

We consider a NN architecture that is used for building and training the baseline models for each of the three sets of time series. The NNs are trained in a cross-learning fashion, using samples from all the time series included in a given set. In many NN applications, deciding the architecture of the models is challenging and hyper-parameters optimization techniques are often employed. However, in this study, we manually set the architecture of the NNs as well as their training process. The reasoning for this decision was threefold. Firstly, the NNs we develop are quite simple and, as a result, it is relatively straightforward to select reasonable values for their hyper-parameters. Secondly, we want to assess and compare the accuracy of individual models and their ensembles using a

**Table 8.1** The architecture and the training hyper-parameters of the baseline NNs used in the ensembles, along with the values selected for the purposes of this study, for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series

<i>Hyper-parameter</i>	M4_Yearly	M4_Quarterly	M4_Monthly
Number of hidden layers	3	3	3
Size of hidden layers	512	512	512
Size of input layer	18	24	54
Size of output layer	6	8	18
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Batch size	512	512	512
Training steps	10,000	10,000	10,000
Loss function	scMAE	scMAE	scMAE

set architecture, without the additional effort and computational cost of back-testing different configurations. Finally, considering different values for major hyper-parameters, such as the input size of the networks and the error measure used during training, is already within the scope of this study anyway. Table 8.1 contains the most critical hyper-parameters and their values for these baseline NNs, for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series.

## RESULTS AND DISCUSSION

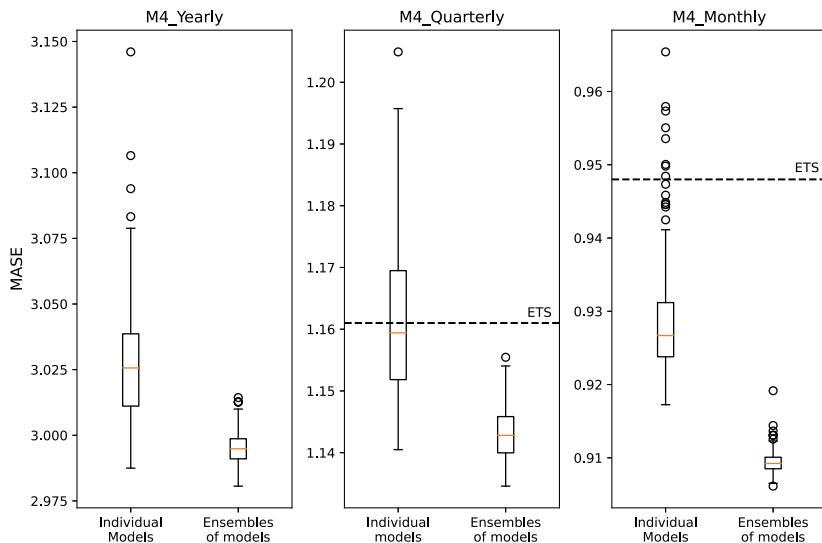
In this section, we report on the results from the experiments conducted, as described in section “[Experimental Design](#).” We first present and discuss the findings concerning each one of the three options for diversifying the base forecasting NNs, i.e., the initialization seed, the training loss function, and the size of the input layer. A final set of experiments investigates the performance of ensembles of NNs with different input layer sizes and training loss functions.

### *Initialization Seed*

The first research question we examined concerns the increases in forecasting accuracy and robustness from simply combining sets forecasts of

NNs that were initialized with different random seeds. Figure 8.1 summarizes the relevant results for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. In each plot, the left candlestick presents the distribution of errors of 200 randomly initialized base NNs, as measured by MASE. The right candlestick presents the distribution of errors of 200 ensembles consisting of 10 randomly initialized base NNs each. The dashed horizontal lines represent the forecasting accuracy of ETS (Hyndman et al., 2002), a strong statistical benchmark, for each set of series. In the case of *M4\_Yearly* the forecasts obtained by ETS have a MASE of 3.444 and the respective line is not depicted as it falls outside the axis range.

The benefits from combining sets of forecasts generated from NN that were initialized and trained with different random seeds are clear for

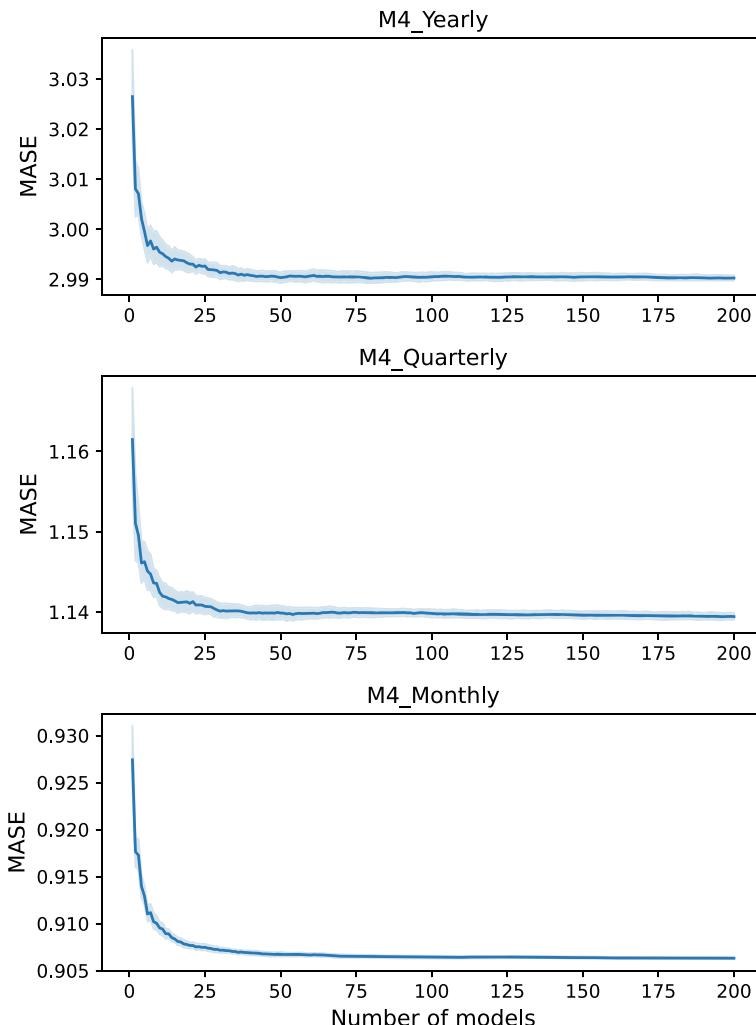


**Fig. 8.1** Distribution of MASE forecasting errors of 200 randomly initialized base NNs models and 200 ensembles ensembles consisting of 10 randomly initialized base NNs each, for the cases of *M4\_Yearly*, *M4\_Quarterly* and *M4\_Monthly* sets of series. The dashed horizontal line represents the forecasting accuracy of ETS. The performance of ETS in *M4\_Yearly* is not shown as it falls outside the axis range (MASE: 3.444)

all three sets of series. First in terms of the expected accuracy, ensembles perform 1.0, 1.6, and 2.2% better than individual models in the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series, respectively. Accuracy, as measured by MASE, improves from 3.027 to 2.996 for the yearly series, from 1.162 to 1.143 for the quarterly series and from 0.929 to 0.909 for the monthly series. The distribution of the measured accuracy across 200 runs is significantly more tight when using ensembles of NNs, with approximately 70 to 90% lower standard deviation compared to single NN runs, from 0.022 to 0.006 for the yearly series, from 0.013 to 0.004 for the quarterly series and from 0.008 to 0.001 for the monthly series. The results in all three sets of series are similar, indicating that even the use of small ensembles can increase forecasting accuracy and robustness. Also when compared to a competitive statistical benchmark such as ETS, NN ensembles generate 13%, 2%, and 4% more accurate forecasts for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. As expected, deploying ensembles consisting of 10 NNs takes 10 times more computational time than deploying a single network. However, in absolute terms, the process can be completed in less than 9 min, even if no parallel processing is used and all NNs are trained sequentially. This means that it is not impractical to consider the use of ensembles for real-life forecasting applications.

Since it is evident that ensembles of neural network can produce more accurate forecasts, we focus on the follow-up question of how many NNs need to be combined in order to maximize these benefits. Figure 8.2 summarizes the relevant findings for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. In each plot, the line represents the mean forecasting accuracy, as measured by MASE, across 20 ensembles of different sizes, ranging from 1 up to 200 models per ensemble.

The reported results paint a similar picture for all three sets of series. As the size of the ensemble increases, the accuracy of the generated forecasts also improves. However, as more and more NNs are added to the ensembles, these improvements diminish. Ensembles of 200 NNs have, on average, an accuracy of 2.990, 1.139, and 0.906, when the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets are examined, respectively. This means that combining more sets of forecasts is expected to improve accuracy by 1.2, 1.9 and 2.3%, for the yearly, quarterly and monthly series, compared to the use of forecasts produced by a single model. At the same time, forecasting performance is more robust, since the standard deviation of the errors is reduced by more than 93% across all three sets



**Fig. 8.2** Forecasting performance (MASE) of ensembles of NNs, for different ensemble sizes, ranging from 1 up to 200 models per ensemble, for the cases of *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series

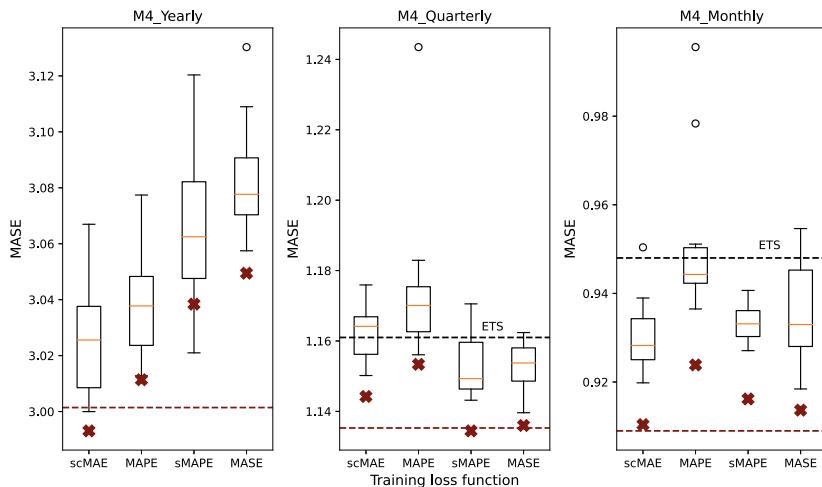
of series, from 0.021% to 0.001% for the yearly series, from 0.014% to 0.001% for the quarterly series, and from 0.008% to less than 0.0005% for the monthly series. Again, it is important to note that in order to maximize forecasting performance, 200 times more computational time is required. Acknowledging the trade-off between increases in accuracy and computational time, smaller ensembles may be more suitable for real-life applications. For example, training and combining 30 NNs realizes approximately 95 to 97% of the potential accuracy improvements in 15% of the time, compared to an ensemble consisting of 200 NNs.

### ***Loss Function***

We examine the accuracy from combining NNs that are trained using different loss functions. The results are summarized in Fig. 8.3 for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* data sets. In each plot, the candlesticks depict the distribution of errors of 10 individual NNs, as measured by MASE, trained with the use of each loss function. The corresponding “X” marks point to the forecasting performance of loss function-specific ensembles consisting of 10 NNs each. The red-dashed horizontal line indicates the forecasting accuracy of the ensemble that utilizes all available NNs, regardless of the function used during training. The black-dashed horizontal lines represent the forecasting accuracy of ETS for each set of series.

The results reported in Fig. 8.3 indicate that the use of broader ensembles is almost always the better choice, regardless of the examined set of series. Ensembling NNs that are trained with the same loss function leads to more accurate forecasts across all loss functions and sets of series, with accuracy improvements ranging from 0.9 up to 2.9%. This is inline with all the aforementioned findings, further reinforcing the argument in favor of NN ensembles. In most cases, creating ensembles that consist of models trained with all four loss functions improves the accuracy of the final forecasts even further.

In the case of *M4\_Yearly*, combining all available models leads to 0.3, 1.2, and 1.6% more accurate forecasts compared to those generated by more limited ensembles consisting of models trained with MAPE, sMAPE, and MASE, respectively. The scMAE-specific ensemble is the only exception, as it is found to be 0.3% more accurate than the broader ensemble. In the case of *M4\_Quarterly* the results are similar, with the more comprehensive ensemble improving accuracy by 0.8, 1.6, and



**Fig. 8.3** Distribution of MASE forecasting errors of 10 base NNs trained with different loss functions (scMAE, MAPE, sMAPE, and MASE), for the cases of *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. “X” marks the forecasting accuracy of the respective loss function-specific ensemble. The red-dashed horizontal line represents the forecasting accuracy of the ensemble that combines the forecasts of all available models, using all four training losses. The black-dashed horizontal line represents the forecasting accuracy of ETS. The performance of ETS in *M4\_Yearly* is not shown as it falls outside the axis range (MASE: 3.444)

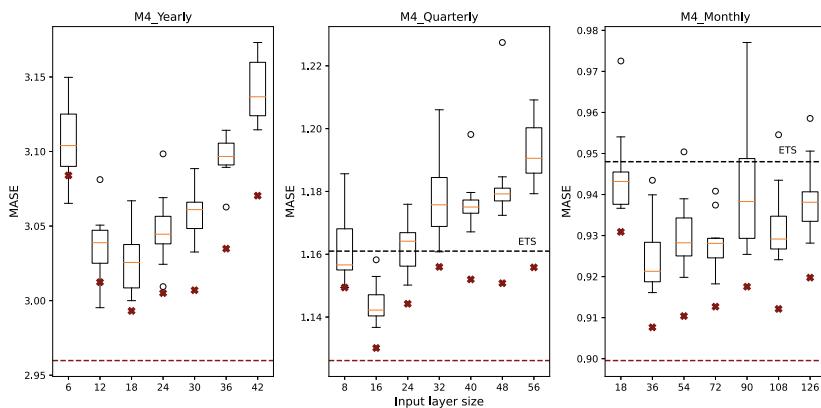
0.1% over the scMAE-, MAPE-, and MASE-specific ensembles. Here, the ensemble of models trained with sMAPE provides 0.1% more accurate forecasts. Finally, while examining the monthly series of *M4\_Monthly*, combining all models, using all four loss functions, increases the accuracy of the forecasts by 0.1, 1.6, 0.8, and 0.5%, when compared to the scMAE-, MAPE-, sMAPE-, and MASE-specific ensembles respectively.

Despite the variations in model performance due to different loss functions used for training, combining all available NNs results in the most accurate or nearly the most accurate forecasts for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. This essentially eliminates the need for identifying the optimal loss function.

### Input Layer Size

The third approach for creating diverse ensembles of models examined in this study is linked to the use of NNs with different input layer sizes. The results are summarized in Fig. 8.4 for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. In each plot, the seven candlesticks show the distribution of errors, as measured by MASE, of 10 individual NNs whose input layer size ranges from 1 up to 7 times the corresponding time series forecasting horizon. The “X” marks point to the forecasting performance of the ensembles comprised of 10 NNs, that share a particular input layer size, each. The red-dashed horizontal line represents the forecasting accuracy of the ensemble that utilizes all available NNs, irrespective of the input size. The black-dashed horizontal lines represent the forecasting accuracy of ETS for each set of series.

The results of Fig. 8.4 suggest that the use of ensembles of NNs increases the overall accuracy of the forecasts, especially when networks



**Fig. 8.4** Distribution of sMAPE forecasting errors of 10 base NNs with different input layer sizes (from 1 up to 7 times the corresponding forecasting horizon), for the cases of *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series. “X” marks the forecasting accuracy of the ensemble combining networks with a particular input layer size. The red-dashed horizontal line represents the forecasting accuracy of the ensemble that combines the forecasts of all available models of different input layer sizes. The black-dashed horizontal line represents the forecasting accuracy of ETS. The performance of ETS in *M4\_Yearly* is not shown as it falls outside the axis range (MASE: 3.444)

with different input layer sizes are combined. Similarly to the previous parts of the study, ensembles consisting of NNs with the same architecture outperform the respective constituent individual networks. The reported accuracy improvements vary depending on the size of the input layer and the set of time series, ranging from 0.7% up to 3.1%. One can observe that while considering different input sizes, a u-shaped pattern emerges for all three sets of series. Input layer sizes, however, may rank differently when evaluated across the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets. Combining the forecasts of all the networks with different input layers seems to be the most beneficial approach in this case too. In the case of *M4\_Yearly*, combining all available models leads to 1.1% to 4.0% more accurate forecasts compared to ensembles of networks with a fixed input layer size. When the series in *M4\_Quarterly* are considered, the broader ensemble outperforms the ensembles of models with fixed input layer sizes by a margin of 0.1% up to 2.2%. Similarly, when forecasting the series in *M4\_Monthly* the final ensemble improves forecasting performance by 0.9 to 3.3% when compared to ensembles of architectures with a particular input layer size.

### *Final Ensemble*

We examine the performance of ensembles that combine NNs with different input layer sizes and training loss functions. The results are summarized in Tables 8.2, 8.3, and 8.4, for the *M4\_Yearly*, *M4\_Quarterly*, and *M4\_Monthly* sets of series.

**Table 8.2** Overall forecasting accuracy (MASE) of ensembles of models with different input sizes (from 1 up to 7 times the corresponding forecasting horizon) and loss functions (sMAPE, MASE, MAPE, scMAE), in the *M4\_Yearly* set of series

<i>Input size</i>	<i>sMAPE</i>	<i>MASE</i>	<i>MAPE</i>	<i>scMAE</i>	<i>All loss functions</i>
6	3.120	3.126	3.118	3.084	3.088
12	3.057	3.096	3.044	3.012	3.029
18	3.038	3.049	3.011	2.993	3.001
24	3.012	3.040	3.012	3.005	2.990
30	3.020	3.034	3.034	3.007	2.996
36	3.017	3.064	3.065	3.035	3.018
42	3.048	3.094	3.104	3.070	3.049
All sizes	2.975	2.990	2.987	2.960	<b>2.959</b>

*Quarterly*, and *M4\_Monthly* sets of series, respectively. Each column corresponds to ensembles of NNs trained with one of the loss functions. Each row corresponds to ensembles of NNs with a specific input layer size. Thus, each cell's value reports on the forecasting accuracy, as measured by MASE, of an ensemble of 10 NNs, with a particular combination of input layer size and training loss function. The last column of the table corresponds to the performances of ensembles that combine all 40 models from each row, while the last row represents the performances of ensembles consisting of all 70 models from each row. The combined sets of forecasts are produced by utilizing the median operator on the forecasts generated by the respective 40 or 70 models participating in each ensemble. The performance of a final ensemble, that leverages all four loss functions and all seven input layer sizes, and consists of 280 NNs, is reported at the bottom right of the table.

The results from Table 8.2 further support the argument in favor of more diverse ensembles of NNs in the case of the *M4\_Yearly* set of series. Combining the forecasts of NNs that are trained with different loss functions can lead to up to 1.8% improvements in forecasting accuracy depending on the size of the input layer, when compared to ensembles of models of a particular configuration. Ensembling NNs with different input sizes provides up to 4.6% more accurate forecasts than combinations of single-configuration networks. Finally, the ensemble consisting of 280 NNs, that mixes all different loss functions and input layer sizes, produces the most accurate forecasts compared to

**Table 8.3** Overall forecasting accuracy (MASE) of ensembles of models with different input sizes (from 1 up to 7 times the corresponding forecasting horizon) and loss functions (sMAPE, MASE, MAPE, scMAE), in the *M4\_Quarterly* set of series

<i>Input size</i>	<i>sMAPE</i>	<i>MASE</i>	<i>MAPE</i>	<i>scMAE</i>	<i>All loss functions</i>
8	1.142	1.144	1.164	1.149	1.143
16	1.128	1.125	1.155	1.130	1.126
24	1.134	1.136	1.153	1.144	1.135
32	1.149	1.143	1.166	1.156	1.146
40	1.144	1.145	1.169	1.152	1.144
48	1.137	1.150	1.161	1.151	1.140
56	1.143	1.153	1.180	1.156	1.148
All sizes	1.117	<b>1.116</b>	1.142	1.126	1.120

**Table 8.4** Overall forecasting accuracy (MASE) of ensembles of models with different input sizes (from 1 up to 7 times the corresponding forecasting horizon) and loss functions (sMAPE, MASE, MAPE, scMAE), in the *M4\_Monthly* set of series

<i>Input size</i>	<i>sMAPE</i>	<i>MASE</i>	<i>MAPE</i>	<i>scMAE</i>	<i>All loss functions</i>
18	0.933	0.924	0.954	0.931	0.926
36	0.907	0.906	0.919	0.908	0.903
54	0.916	0.914	0.924	0.910	0.909
72	0.917	0.921	0.920	0.913	0.910
90	0.924	0.925	0.924	0.918	0.916
108	0.917	0.919	0.928	0.912	0.911
126	0.925	0.923	0.923	0.920	0.917
All sizes	0.904	0.902	0.910	0.900	<b>0.899</b>

all other approaches, with increases in accuracy ranging from 0.1 to 5.3%. However, it is important to note that the computational time needed for deploying such a large ensemble is considerable. It takes more than 3 h to train and use all 280 models, and, if the ability to develop NNs in parallel is not available, these performance improvements may be out of reach in certain scenarios.

A similar picture emerges from Tables 8.3 and 8.4, concerning the *M4\_Quarterly* and *M4\_Monthly* sets of series. When NNs trained with all four different loss functions are combined, forecasting accuracy improves by an average of 0.7 and 0.8% for the quarterly and monthly series, respectively, compared to ensembles consisting of networks trained with a single loss function. Likewise, combining the forecasts of NNs with different input layers sizes leads, on average, to 2.0% and 1.8% more accurate results, when evaluated on the *M4\_Quarterly* and *M4\_Monthly* set of series. The final ensemble, which combines all 280 available models, produces forecasts in the quarterly set of series that are approximately 2.1% more accurate than the rest of the ensembles. However, it is worth noting that despite being the third best performing approach, it is only 0.3 and 0.4% less accurate than the top two performing configurations. In the *M4\_Monthly* data set, the broader ensemble provides the most accurate forecasts when compared to all other approaches, with performance improvements ranging from 0.1 to 5.8%. Similarly to the case of *M4\_Yearly*, computational time is again an important consideration, since both ensembles require close to 4 h in order to generate the final forecasts.

## CONCLUSIONS

The present study examined the improvements in forecasting accuracy from employing ensembles of NNs in the context of a univariate time series forecasting problem. We focused on combining the forecasts of shallow, feed-forward, and fully connected networks by using the simple median operator. In order to create diverse ensembles of models, we considered changes in the values of three key hyper-parameters that define the topology and training of NNs, namely the initial seed provided for building and training NNs, the size of the input layer of the networks, and the loss function used during training. The performance of individual models and their respective ensembles for different sets of hyper-parameters were empirically evaluated using three large data sets consisting of series of different frequencies.

The results of the present study strongly support the use of ensembles of NNs instead of individual models. Even when using the median operator for combining the forecasts from 10 randomly initialized NNs, forecasting accuracy increases up to 2.2%. Note that, although an increase of 2.2% may seem low, one must consider that even individual NNs are among the most accurate forecasting models available. At the same time, the overall performance is up to 90% more robust, with identical ensembles generating forecasts of similar accuracy. Also, these improvements can be easily realized in real-life applications since deploying these ensembles requires less than 10 minutes.

A second significant finding is that considering pools of models with different values of specific hyper-parameters, such as the size of the input layer and the loss function used for training, reduces the need for identifying the “optimal” set of hyper-parameters. In the majority of cases, these ensembles, based on NNs with different configurations, perform better than individual models or ensembles of models with a particular configuration. Even in the few instances where this is not the case, the broader ensemble’s performance is highly comparable with that achieved by the best setup. As a result, it is possible to avoid the challenges and computational resources needed for determining the optimal set of hyper-parameters for a particular problem, without compromising the accuracy of the final forecasts.

The potential trade-offs between accuracy improvements and increased computational requirements for deploying ensembles of multiple NNs need to be discussed. It is clear that as more models are added to an

ensemble, the computational time required for training and employing them linearly increases. For example, a single NN can be trained to forecast all series in *M4\_Monthly* in less than a minute, whereas training and obtaining forecasts from all 280 models in the final ensemble takes more than 4 hours. Although such a 280-times increase in computational time seems extreme at first glance, one has to consider the particular forecasting scenario to determine whether such a cost is actually prohibitive. Firstly, ensembling a large number of NNs with various configurations significantly reduces the need for hyper-parameter optimization or eliminates it altogether. It is a time-consuming and computationally expensive task to set up a validation process and iterate through different hyper-parameter values. Deploying an ensemble that consists of all models configurations without exhaustively back-testing them may be a simpler and more efficient solution. Secondly, it is important to consider whether models should be re-trained often in a particular forecasting scenario. For instance, in monthly, quarterly, or yearly series forecasting applications, re-training models once or twice a year is reasonable, and a 4-hour training process is not prohibitively expensive or impractical. However, if the forecasting task requires daily re-training, then 4 hours may be too long. Finally, while larger ensembles examined in this study take 4 hours to train, more limited ensembles can generate comparably accurate forecasts in a fraction of the time. In most scenarios, a total model training time of less than an hour would be acceptable, especially as it can be further reduced by utilizing more computational resources and parallelism in training.

The findings of this study motivate further research on creating diverse ensembles of NNs for univariate time series forecasting. Some promising aspects that could be examined in the future are listed below:

- In this study, we focused on combining the forecasts of simple, feed-forward, and fully connected NNs. It would be interesting to investigate ensembles consisting of additional types of networks (e.g., recurrent and convolutional networks), in order to assess their performance in terms of accuracy and robustness.
- Similarly, the current experimental design considered three aspects that influence the forecasts generated by each model, i.e., the initial random seed, the size of the input layer, and the training loss function. In the future, the empirical evaluation can be expanded to include more high-level network hyper-parameters (e.g., the size of

the output layer, the number and size of the NN layers) in order to produce and combine more diverse sets of forecasts.

- Finally, the combination scheme used in this paper relies on combining all possible configurations in the final ensemble. Although this naive approach improves forecasting performance, it requires significant computational time. Additional, smarter strategies for adding models in an ensemble can be reviewed, in order to increase accuracy and reduce computational time.

## REFERENCES

- Ahmed, H., & Lofstead, J. (2022). Managing randomness to enable reproducible machine learning. In *Proceedings of the 5th International Workshop on Practical Reproducible Evaluation of Computer Systems*, Association for Computing Machinery, New York, NY, USA, P-RECS '22, pp. 15–20.
- Aiolfi, M., & Timmermann, A. (2006). Persistence in forecasting performance and conditional combination strategies. *Journal of Econometrics*, 135(1), 31–53.
- Barker, J. (2020). Machine learning in M4: What makes a good unstructured model? *International Journal of Forecasting*, 36(1), 150–155.
- Bates, J. M., & Granger, C. W. J. (1969). *The combination of forecasts*. *OR*, 20(4), 451–468.
- Choi, J. Y., & Bumshik, L. (2018). Combining LSTM network ensemble via adaptive weighting for improved time series forecasting. *Mathematical Problems in Engineering*, 2018, 8.
- Gastinger, J., Nicolas, S., Stepić, D., Schmidt, M., & Schülke, A. (2021). A study on ensemble learning for time series forecasting and the need for meta-learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp 1–8.
- Godahewa, R., Bandara, K., Webb, G. I., Smyl, S., & Bergmeir, C. (2021). Ensembles of localised models for time series forecasting. *Knowledge-Based Systems*, 233, 107518.
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688.
- Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18(3), 439–454.
- Kang, Y., Cao, W., Petropoulos, F., & Li, F. (2022). Forecast with forecasts: Diversity matters. *European Journal of Operational Research*, 301(1), 180–190.

- Kourentzes, N., Barrow, D. K., & Crone, S. F. (2014). Neural network ensemble operators for time series forecasting. *Expert Systems with Applications*, 41(9), 4235–4244.
- Kourentzes, N., Barrow, D., & Petropoulos, F. (2019). Another look at forecast selection and combination: Evidence from forecast pooling. *International Journal of Production Economics*, 209, 226–235, the Proceedings of the 19th International Symposium on Inventories.
- Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10), 2006–2016, subspace Learning/Selected papers from the European Symposium on Time Series Prediction.
- Li, H., Tang, M., Liao, K., & Shao, J. (2022). A multi-output integration residual network for predicting time series data with diverse scales. In S. Khanna, J. Cao, Q. Bai, & G. Xu (Eds.), *PRICAI 2022: Trends in Artificial Intelligence* (pp. 380–393). Springer Nature.
- Makridakis, S. (1993). Accuracy measures: Theoretical and practical concerns. *International Journal of Forecasting*, 9(4), 527–529.
- Makridakis, S., & Hibon, M. (2000). The m3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4), 451–476, the M3-Competition.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4), 802–808.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74.
- Makridakis, S., Spiliotis, E., Assimakopoulos, V., Semenoglou, A. A., Mulder, G., & Nikolopoulos, K. (2022). Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward. *Journal of the Operational Research Society*, 74(1), 1–20.
- Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFOMA: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1), 86–92.
- Narkhede, M. V., Bartakke, P. P., & Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, 55(1), 291–322.
- Oreshkin, B. N., Carpor, D., Chapados, N., & Bengio, Y. (2019). *N-BEATS: neural basis expansion analysis for interpretable time series forecasting*. CoRR abs/1905.10437, 1905.10437
- Petropoulos, F., & Svetunkov, I. (2020). A simple combination of univariate models. *International Journal of Forecasting*, 36(1), 110–115, m4 Competition.

- Petropoulos, F., Hyndman, R. J., & Bergmeir, C. (2018). Exploring the sources of uncertainty: Why does bagging for time series forecasting work? *European Journal of Operational Research*, 268(2), 545–554.
- Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85.
- The M4 Team. (2018). *M4 Competition methods*. <https://github.com/M4Competition/M4-methods>, GitHub repository available at <https://github.com/M4Competition/M4-methods>
- Wang, X., Hyndman, R. J., Li, F., & Kang, Y. (2022). *Forecast combinations: An over 50-year review*. 2205.04216
- Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160(2), 501–514.

---

PART IV

---

## Meta-Learning and Feature-Based Forecasting



## CHAPTER 9

---

# Large-Scale Time Series Forecasting with Meta-Learning

*Shaohui Ma and Robert Fildes*

## INTRODUCTION

Forecasting time series is essential in many contexts. Since the 1950s, time series forecasting has been a very active area of research. A large number of forecasting models have been developed. According to the “No Free Lunch” theorem of Wolpert & Macready: “there is no guarantee that any method, no matter how complex it is, will always perform better than another prediction method for different datasets” (Wolpert & Macready, 1997); in the time series forecasting domain, this implies that it is unlikely

---

S. Ma

School of Business, Nanjing Audit University, Nanjing, China  
e-mail: [shaohui.ma@nau.edu.cn](mailto:shaohui.ma@nau.edu.cn)

R. Fildes (✉)

Centre for Marketing Analytics and Forecasting, Lancaster  
University,  
Bailrigg, UK  
e-mail: [r.fildes@lancaster.ac.uk](mailto:r.fildes@lancaster.ac.uk)

that one single model will dominate others for all of a set of time series and all the periods.

Traditionally, how to select a forecasting model given a time series usually relied heavily on the forecaster's experience. A skilled analyst can forecast a single time series by applying good judgment based on his or her knowledge and experience, using various time series analysis techniques, and utilizing good software based on proven statistical theory. However, in the big data era, many applications are concerned with forecasting large sets of time series. Consider, for example, in transportation, where real-time (15–40 min) forecasting of traffic flow gives travelers the ability to choose better routes and the traffic administrator the ability to manage the transportation system. In the retail industry, retailers need to forecast the daily demand of thousands or even millions of products across thousands of locations. Selecting models for predicting large numbers of time series requires some degree of automation. Meta-learning aims to develop meta-knowledge on model selection or combination from past predictive tasks. The meta-knowledge can then recommend the proper forecasting method for a time series according to its data characteristics without human intervention. Though meta-learning incurs extra computing cost due to training multiple base forecasters on a large repository of historical time series (usually many more time series than those in the forecasting task at hand), it is an offline procedure. Once the meta-knowledge on model selection/ensemble is obtained, in the meta-forecasting stage, the computing cost is similar to that of traditional model combination.

In this chapter, we will briefly review the current development of meta-learning methods in time series forecasting, summarize a general meta-learning framework for time series forecasting, and discuss the key elements of establishing an effective meta-learning system. We will also introduce a meta-learning class library that we developed based on the Python and PyTorch deep-learning frameworks. This class library can make it easier for researchers and practitioners to implement meta-learning in large-scale time series forecasting tasks and develop new meta-learning components within the framework. Further research on meta-learning will also be discussed at the end of this chapter.

## A BRIEF REVIEW OF THE META-LEARNING IN TIME SERIES FORECASTING

In computer science, meta-learning is commonly referred to as ‘learning to learn’. The primary goal of meta-learning is understanding the interaction between the mechanism of learning and the concrete contexts in which that mechanism is applicable (Giraud-Carrier, 2008). In general, a meta-learning algorithm is trained with outputs (i.e., the model’s predictions) and metadata of machine learning algorithms to extract and transfer knowledge that guides the search for optimal models for new tasks. The metadata may comprise the measurable properties of the task itself and the exact algorithm configurations used to train the models. Meta-learning helps researchers understand which algorithm(s) generate the best/better predictions from datasets.

The idea of meta-learning is not new; one of the first and seminal contributions was provided by Rice (1976). However, meta-learning studies became popular after Jürgen Schmidhuber and Yoshua Bengio’s works on the topic (Bengio et al., 1991; Schmidhuber, 1992). Since then, meta-learning has been applied to algorithm recommendation (including classification, regression, ranking, constraint satisfaction, and optimization), model combination (Bonissone, 2012), parameter initialization (Reif, Shafait, & Dengel, 2012), hyper-parameter tuning (Molina, Romero, Ventura, & Luna, 2012), dynamic bias selection (Kadlec & Gabrys, 2008), and inductive transfer (Aiolli, 2011).

In recent years, as more and more real-life problems are solved using artificial intelligence (AI) algorithms, the challenges when working with AI on selecting and tuning the best algorithm for solving the problem in hand have increased interest in meta-learning studies. Meta-learning in recent years has been applied to Denil, M. optimizing deep networks (Andrychowicz et al., 2016; Li & Malik, 2017), learning dynamically changing recurrent networks (Ha, Dai, & Le, 2017), as well as learning both the weight initialization and the optimizer for few-shot image recognition (Ravi & Larochelle, 2017). For more detail on the general meta-learning literature, readers can refer to survey papers published in recent years (e.g., (Hospedales, Antoniou, Micaelli, & Storkey, 2022; Ma et al., 2021; Wang, 2021)).

In time series forecasting research, the idea of using characteristics of univariate time series to select an appropriate forecasting model has been pursued since the 1990s. The first systems were rule based and built on a mix of judgmental and quantitative methods. Collopy and Armstrong (1992) used time series features to generate 99 rules for weighting four different models; features were obtained judgmentally by visually inspecting the time series and using domain knowledge. Such rule-based methods were extensions of the fundamental statistics model selection approach of classifying a time series by its components identified by decomposition, and matching to an appropriate model or combination. Ultimately, automatic systems have been proposed by Arinze, Kim, and Anandarajan (1997), where a generated rule base selects between six forecasting methods. Discriminant analysis to choose between three forecasting methods using 26 features was used by Shah (1997). Meade (2000) found that summary statistics that summarize the data available in a time series can be used to select a good forecasting method (or set of methods) but not necessarily the best.

The term meta-learning was first adopted in forecasting many time series by Prudêncio and Ludermir (2004). They presented two case studies. In the first one, they used ten self-defined features and two base forecasters (the simple exponential smoothing model (SES) and the time-delay neural network (TDNN)). They used a C4.5 decision tree as the meta-learner to learn the rules from 99 time series on when one base forecaster performs better than the other. Their out-of-sample test results indicated that using the model recommended by the meta-learner, on average, provides more accurate forecasts than using either of the base forecasters as the default model. In the second case, they used an artificial neural network as the meta-learner, to predict the forecasting performance ranking of three base forecasters based on M3 competition data. However, they did not compare the performance of the two meta-learners. Wang, Smith-Miles, and Hyndman (2009) used a more comprehensive set of features and used both supervised (decision tree) and unsupervised (self-organizing map, SOM) meta-learners, provided both rules, as well as visualizations in the feature space. They found that a derived weighting schema based on the rule induction to combine base forecasters could further improve forecasting accuracy over only using one

of the base forecasters. Similarly, Lemke and Gabrys (2010) showed the superiority of a ranking-based combination with base forecasters over the standard approach of selecting a single model. Widodo and Budi (2013) proposed to reduce the dimensionality of features extracted from time series by Principal Component Analysis before they are used for meta-learning. They found that the dimensionality reduction might help the meta-learner to find the appropriate model. Fildes and Petropoulos (2015) found accuracy benefits from selecting a range of methods to match the data and series forecasting performance characteristics, which suggests there may be benefits to be had from extending this approach to specific applications such as retailing and including a wider range of methods and series features that capture the characteristics of particular application. Kück, Crone, and Freitag (2016) proposed a meta-learner based on neural networks. They introduced a new set of features based on the forecasting errors of a set of commonly used forecasting methods, building on the core forecasting concept of using past performance as the key predictor of future performance. They showed promising results of including error-based feature sets in meta-learning for selecting time series forecasting models. Cerqueira, Torgo, Pinto, and Soares (2017) proposed a meta-learner to predict the absolute error of each base forecasters and then use the predicted error as the performance measurements to ensemble base forecasters with a softmax function. The method provided more accurate forecasts than the weighted linear combination, but it was not compared with other meta-learners, such as model selecting and ranking. Similarly, Talagala, Li, and Kang (2022) forecast the forecast error of a set of base forecasters with a Bayesian multivariate surface regression approach using time series features calculated from historical time series. The approach has a lower computational cost and a higher degree of interpretability.

Talagala, Hyndman, and Athanasopoulos (2018) proposed a meta-learning framework named FFORMS (Feature-based FORecast Model Selection) that uses a set of 25 features for non-seasonal data and 30 features for seasonal data, and then use Random Forest as a meta-learner to select the best single forecasting model from nine base forecasters. To build a reliable classifier, they proposed to augment the set of observed time series by simulating new time series similar to those in the assumed population. Montero-Manso, Athanasopoulos, Hyndman, and Talagala

(2018) built on FFORMS by using meta-learning to select the weights for a weighted forecast combination. All base forecasters are applied, and the weights to be used in combining them are chosen based on the features of each time series. They call this framework FFORMA (Feature-based FORest Model Averaging). FFORMA resulted in the second most accurate point forecasts and prediction intervals among all competitors in the M4 competition (Makridakis & Petropoulos, 2019), which led to the extensive attention given to meta-learning methods in time series forecasting in recent years. One of the drawbacks of FFORMA is that it targets minimizing the combined loss during the training, not the loss of combined forecasts directly, which can result in suboptimal combinations. Another issue is that it depends on judgmental and predefined features, but it is a difficult task to capture intrinsic properties embedded in various time series data when the feature space is large.

Ma and Fildes (2021) proposed a meta-learning framework based on deep convolutional neural networks; these can first learn a supervised feature representation from the raw time series and any available influential factors automatically, and then link these learned features with a set of weights which in turn weight the base forecasters to deliver a combined forecast. Their experiments on IRI retail weekly sales data showed that their proposed meta-learner provided superior forecasting performance compared to several state-of-art benchmarks.

## KEY CHOICES IN DESIGNING AN EFFECTIVE META-LEARNING FRAMEWORK

A general meta-learning framework for large-scale time series forecasting is presented in Fig. 9.1. Implementing the framework consists of two phases: meta-learning and meta-forecasting. In the meta-learning phase, forecasters first need to prepare a large time series repository containing both the forecasted time series and the external influential factors. The time series in the repository should be similar to those that will be forecasted. In some application areas, such as retail, transportation, and energy, it is usually easy to accumulate massive amounts of directly relevant historical time series data. Even if additional similar time series are not easy to collect, an option is simply augmenting the time series to be forecasted using a rolling window strategy to build the repository.

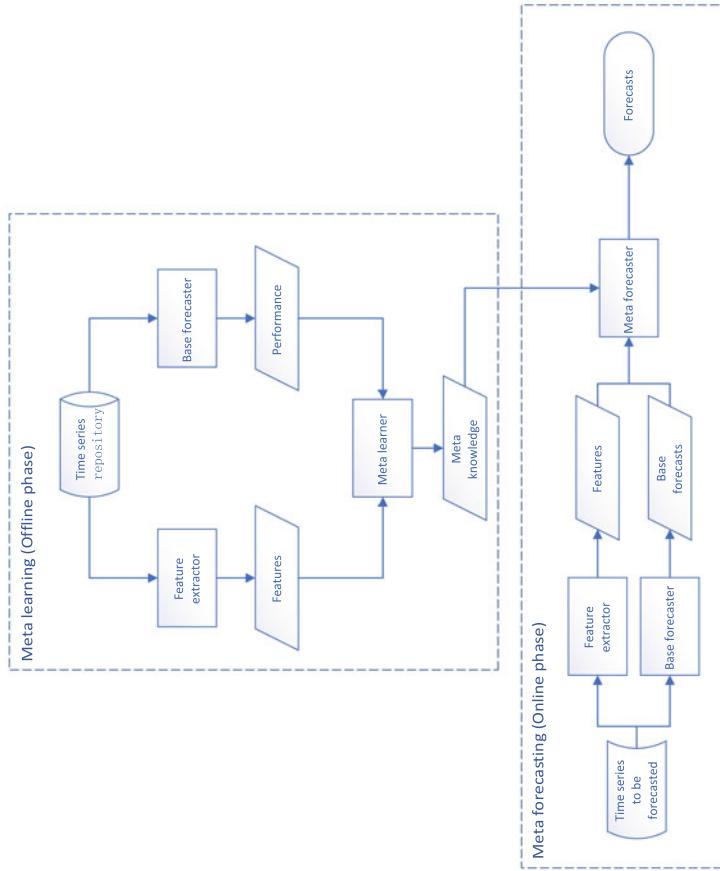


Fig. 9.1 A general meta-learning framework for time series forecasting

Meta-learning works on metadata that contains knowledge about earlier learning experiences. In the context of time series forecasting, the metadata mainly consists of three parts: the features of the time series, the forecasting performance of the base forecasters on the same set of time series, and the training configurations of those base forecasters. The meta-learner is used to extract meta-knowledge that links the features of the time series with the performance of base forecasters under specific configurations. During the meta-forecasting phase, the meta-forecaster uses the meta-knowledge to select, rank, or ensemble base forecasters to generate the final forecasts.

### *Selection of Base Forecasters*

Generally, any time series forecasting model can be used as a base forecaster. However, as many models have been developed, the selection of base forecasters still requires a researcher with rich forecasting experience and domain knowledge and overall performance depends on a suitable choice set (Ma & Fildes, 2020). Usually, the selection of the base forecasters depends much on the characteristics of the historical time series data (seasonality, trend, and intermittency, etc.), the availability of additional explanatory variables, the features of the prediction task (number of time series, length of each time series, the correlation between individual series, or length of prediction horizon), etc. (Fig. 9.1).

In general, the selection of the base forecasters for meta-learning should follow three rules: (1) we should select classical and robust forecasting models in the forecasting task domain; (2) each base forecaster should perform well for some relevant scenarios; (3) more base forecasters could be included if we have a large enough time series repository for meta-learning.

The modeling strategies for large-scale time series forecasting problems can be classified into two extremes: modeling each time series individually or modeling the whole set of time series in a pooled fashion (Ma & Fildes, 2020). Januschowski et al. (2020) distinguished the two strategies as local and global forecasting methods. Local forecasting methods treat each time series separately and forecast them in isolation, which can fully consider each time series' data characteristics, such as seasonality

and temporal trend. Most traditional statistical forecasting methods have been developed mainly in this setting. Simple moving averages, the exponential smoothing family or Autoregressive Integrated Moving Average (ARIMA), are among of most popular local forecasting methods. Local methods cannot capture many data features common among different time series due to data scarcity; this is especially true for short time series with high noise (Duncan, Gorr, & Szczypula, 2001). The global strategy, on the other hand, can learn the common patterns across the set of available time series due to sufficient data are available, and thus can consider a vast number of dynamic features to improve the prediction accuracy. However, the disadvantage is that the global approach tends to ignore the individual characteristics of the time series.

Therefore, if the set of time series to be predicted comes from different domains, each time series is independent of each other and lacks common dynamic patterns, then a base forecaster pool consisting of only local forecasting models may work well. Otherwise, if many related time series are forecasted, global forecasting models should be included in the base forecaster pool. Ma and Fildes (2021) used 11 base forecasters for the SKU-level retail demand forecasting, including both global and local forecasting models. Their empirical results showed that the two models have their strengths for various time series. The meta-learners using a mixed pool of base forecasters improve the forecasting performance significantly over meta-learners using base forecasters consisting of only local or global forecasting models.

### *Feature Extracting*

Features describing the prediction tasks are one of the most important metadata for meta-learning. In time series forecasting, we can regard the forecasting of a single time series or the forecasting of a group of time series as a forecasting task. There have been many studies and tools for feature extraction from time series. For example, the R package ‘tsfeatures’ by Hyndman et al. (2019) can extract tens of features from a time series, including trend, seasonality, linearity, spikiness, lumpiness, entropy, etc. Montero-Manso et al. (2018) used 42 features extracted with the ‘tsfeatures’ package in their FFORMA model. Package ‘theft’ (Henderson, 2023) and ‘feasts’(O’Hara-Wild, Hyndman, & Wang, 2022) also provide tools for handling the extraction of features from time series. There are also python libraries that provide similar functions, e.g.,

‘TSFEL’ by Barandas et al. (2020) and ‘tsfresh’ by Christ, Braun, Neuffer, and Kempa-Liehr (2018); the latter by default can compute a total of 794 time series features.

Though many existing tools facilitate the feature-extracting process, it is difficult to determine useful features only judgmentally to capture the intrinsic properties embedded in various time series data when the feature space is large. Researchers have to consider the multifarious algorithms of signal processing and time series analysis for identifying and extracting meaningful features from time series. The workload from the calculation of these features is also substantial. Instead of judgmental feature extraction, Ma and Fildes (2021) used convolutional neural networks to extract time series features and integrated the feature extractor with the meta-learner into one neural network, which can extract supervised features from time series automatically. This simplifies the data processing effort and makes the meta-learning approach easy to implement. Their empirical study found that the meta-learner using a supervised feature-extracting network consistently performed better than the meta-learner using unsupervised hand-selected features.

When time series of various potentially influential factors are also available, extracting features from both the target time series and their influential factors may be necessary. In the context of SKU-level sales forecasting in retail, sales are affected significantly by several factors, such as price reduction, aisle display, feature advertising, and calendar events. These factors are usually known in advance and typically cover the forecasting period. Ma and Fildes (2021) found that the meta-learner, which extracted features from both the time series of sales and its influential factors, improved forecasting performance over the meta-learner using only features extracted from the sales time series.

### *Meta-Learners*

To ‘learn how to learn’, a learning algorithm is essential. The meta-learner here is used to learn the best forecasting methods for a particular time series. According to the objective of a meta-learner, we can classify meta-learners into three categories: model selection, loss prediction, and model combination.

The meta-selection learner aims to select the best forecasting model among a set of base forecasters for a time series according to its features. So the training of a meta-selection learner is a classification problem in machine learning. Random forest or gradient boosting trees are usually the best choices to work as model selection learners when using judgmentally selected features. When using a neural network as the supervised feature extractor, a neural network with a softmax output layer should be used as a meta-selection learner to train both networks jointly with a joint loss function.

The loss prediction learner aims to predict the forecasting error of each of the base forecasters. The training of such a meta-learner is a regression problem. Like the meta-selection learner, random forest, gradient boosting trees, or deep-learning neural networks are usually chosen to work as loss prediction learners. During the meta-forecasting phase, the trained meta-learner generates loss predictions for each base forecaster, and the loss predictions are then used to select the best base forecaster or as weights to form a combined forecast.

According to the features of a time series to be forecasted, the model combination learner aims to learn a set of optimal weights, which combines base forecasters to provide the best forecasts for the time series. Usually, only machine learning models such as neural networks that can generate multivariate outputs (i.e., weights) can be used as the model combination learner.

Ma and Fildes (2021) compared the forecasting performance of the three categories of meta-learners and found that the forecasting accuracy of the model combination learner was significantly higher than that of the other two. However, model selection or combination learning also suffers limitations: adding a new or dropping an existing forecaster requires the meta-learner's re-training, which causes considerable difficulty when maintaining large meta-learning libraries. In contrast, the loss prediction learner has greater flexibility: it is possible to train a meta-learner for each base forecaster separately, so as to establish a growing meta-learning library without needing to update the meta-learner frequently, which is valuable for the establishment of a large-scale meta-learning library.

## A PYTHON LIBRARY FOR TIME SERIES FORECASTING WITH META-LEARNING

We developed an open-source Python class library called ‘tsfmeta’ for time series forecasting with meta-learning<sup>1</sup>. The library’s goal is to make meta-learning available for researchers and time series forecasting practitioners in a unified, easy-to-use framework. ‘tsfmeta’ was created with foundations on existing libraries in the PyTorch eco-system, streamlined neural network layer definitions, temporal snapshot generators for batching, and integrated benchmark datasets. Specifically, the library provides a metadata class that abstracts handling data initialization, data iterating, train-test splitting, and feature extracting, etc., a meta-learner class that provides multiple predefined neural network architectures for meta-learning, and a utility class that provides functions that help users to create metadata with their datasets.

### *Raw Data Class*

Raw data class is an interface to facility users preparing metadata with their raw time series data. Assuming users’ custom time series dataset is stored in a pandas DataFrame of long format as the following sample extracted from the IRI retail dataset.

```
iridata = pd.read_csv('test_data.csv')
iridata
```

<sup>1</sup> Code and documentation are available in <https://github.com/Shawn-nau/tsfmeta>.

	IRI_KEY	SY	GE	VEND	ITEM	cate	week	units	price	D	F	holiday
0	200655	0	1	14054	9605	4	1	6	2.896667	0	0	0
1	200655	0	1	18200	466	0	1	30	1.660000	0	0	0
2	200655	0	1	18200	468	0	1	22	1.660000	0	0	0
3	200655	0	1	18200	769	0	1	9	8.990000	0	0	0
4	200655	0	1	18200	53030	0	1	2	15.990000	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
381695	6001821	0	4	94514	46	2	55	6	2.941667	0	0	1
381696	6001821	0	6	25500	20012	1	55	4	9.562500	0	0	1
381697	6001821	6	1	35985	60611	0	55	12	6.790833	0	0	1
381698	6001821	7	1	18200	53025	0	55	70	14.480143	1	0	1
381699	6001821	7	1	42365	25425	4	55	3	3.783333	0	0	1

381700 rows × 12 columns

The ‘IRI\_KEY’ is the masked store number; ‘SY’, ‘GE’, ‘VEND’, and ‘ITEM’ are the UPC (Universal Product Code) fields; ‘cate’ represents the code of product category; ‘week’ is the IRI week; ‘units’ is the product sales which is the target to be forecasted; ‘D’ and ‘F’ are the indicators of aisle display (1 if true, else 0) and feature advertising (1 if true, else 0), respectively. We provide a transform function ‘Df\_to\_rawdata’ to transform the DataFrame data into a Raw data class object once given the column name of the time series indexes, time indexes, target, and the features used to forecast the target. Those features can be further classified into numeric and categorical features. Users also need to indicate which features are known during the forecasting periods, and which are static. Following demonstrative codes transform the IRI DataFrame into a rawdata object.

```
from tsfmeta.data import Md_utils
target = ['units']
item_idx = ['IRI_KEY', 'SY', 'GE', 'VEND', 'ITEM']
time_idx = ['week']
features_num = ['price']
features_cat = ['D', 'F', 'holiday', 'cate']
dynamic_known_features_num = ['price']
dynamic_known_features_cat = ['D', 'F', 'holiday']
static_known_features = ['cate']
df = Md_utils.Df_to_rawdata(df = iridata,
                            target = target,
                            idx = item_idx,
                            tdx = time_idx,
                            features_num = features_num,
                            features_cat = features_cat,
                            static_known_features =
                            static_known_features,
                            dynamic_known_features_num =
dynamic_known_features_num,
                            dynamic_known_features_cat =
dynamic_known_features_cat, )
```

Once a rawdata object is instantiated, we can augment the data into several rawdata slots with the data rolling technique given the rolling window width  $W$  and forecasting horizon  $H$  with class function ‘**data\_segment()**’. All the output data slots are stored in a list object.

```
meta_slots = df.data_segment(W= 48 , H = 7)
```

For each slot of rawdata object, we then transform it into a metadata object with in-class function ‘**Raw2Meta()**’.

```
df_slots = [df_slots[i].Raw2Meta(H = 7) for i in range(len(df_slots))]
```

### *Metadata Class*

The metadata class defines data structures that can efficiently store the datasets and provide temporally ordered snapshots for batching. Its goal is to encapsulate all the metadata and define per-sample loading. Let.

- N be the number of time series;
- T be the maximum length of the time series;
- H be the maximum forecasting horizon;
- M be the number base forecasters;
- C be the number of influential time series;
- K be the number of predefined features extracted from each time series.

To instantiate a metadata object, we need the following inputs,

- X - NumPy Array of Shape (N, C + 1, T) Representing Time Series to Be Forecasted and Its Influential Factors;
- Z - (optional) NumPy array of shape (N, C, H) representing the influential factors during the forecasting periods;
- B - (optional) NumPy array of shape (N, M, H) representing the base forecasts;
- F - (optional) NumPy array of shape (N, K) representing the time series features;
- Yc - (optional) NumPy array of shape (N, M) representing the labels of the best base forecaster, which is used for training the model selection learner;
- Ye - (optional) NumPy array of shape (N, M) representing the loss of the base forecasts, which is used for training the loss prediction learner;

- $Y$  - (optional) NumPy array of shape  $(N, H)$  representing the actual value of the time series forecasting, which is used for training the model combination learner and for calculating  $Y_c$  and  $Y_e$ .

Assume we have already had at least the data array  $X$ ,  $Y$ , and  $Z$ , with the ‘tsfmeta’, we can define a metadata object simply with the following codes,

```
from tsfmeta.data import MetaDataset
df_meta = MetaDataset (X = X, Y = Y, Z = Z)
```

Alternatively, we can also instantiate a metadata object with the function ‘**Raw2Meta()**’ we demonstrated in the Raw Data Class section.

With a metadata object, we can directly generate local base forecasts based on data array  $X$  given the forecasting horizon  $H$ . For example, the following codes generate base forecasts in each metadata slot with simple exponential smoothing.

```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
for slot in range(len(df_slots)):
    Nts, _, T = df_slots[slot].X.shape
    _, H = df_slots[slot].Y.shape
    model = [SimpleExpSmoothing(df_slots[slot].X[i, 0]).fit() for i in range(Nts)]
    model_pred = [model[i].predict(T, T + H - 1) for i in range(Nts)]
    model_pred = np.stack(model_pred, 0)
    df_slots[slot].add_Base_forecasts(model_pred, forecaster='SimpleExpSmoothing')
```

After the forecasts are generated, the forecasts can be added to the base forecaster pool  $B$  in the metadata object with the class function ‘**add\_Base\_forecasts()**’.

For facilitating the training of global forecasting models, we provide an in-class data transform function ‘**XZ\_globalReg()**’. Given the number of lags to work as the explanatory features and the time steps for the data rolling, this function transforms the time series array  $X$  and influential series  $Z$  into a set of data arrays  $[X_{train}, y_{train}, X_{test}]$ . Data arrays  $X_{train}$  and  $y_{train}$  are used to train global forecasting models, such as gradient boosting trees and random forest, and  $X_{test}$  is used to generate base forecasts. The following codes return a python Dictionary object which

contains all the data to train a global forecasting model and the test data to generate the base forecasts.

```
gdf = df_slots[0].XZ_globalReg(L=7, rolling_step = 1, mode='Tree')
gdf.keys()
Output: dict_keys(['X_tr', 'Y_tr', 'X_ts'])
```

We can train a global boosting tree model using LightGBM (Ke et al., 2017) and generate another base forecast with the following demonstrative codes.

```
# lightgbm example
import lightgbm as lgb
from sklearn.model_selection import train_test_split
for slot in range(len(df_slots)):
    gdf = df_slots[slot].XZ_globalReg(L=7, rolling_step = 1, mode='Tree')
    H = gdf['Y_tr'].shape[-1]
    X_train, X_val, y_train, y_val, = train_test_split(
        np.concatenate([gdf['X_tr'], gdf['Z_tr']], -1),
        gdf['Y_tr'],
        test_size = 0.2,
        random_state=0)
    predicts = []
    for h in range(H):
        dtrain = lgb.Dataset(X_train , label= y_train[:,h])
        dval = lgb.Dataset(X_val , label= y_val[:,h])
        params = {
            'num_leaves': 128,
            'objective': 'regression',
            'min_data_in_leaf': 10,
            'learning_rate': 0.02,
            'feature_fraction': 0.8,
            'bagging_fraction': 0.7,
            'bagging_freq': 1,
            'metric': 'rmse',
            'num_threads': 8
        }
```

```

bst = lgb.train(params, dtrain, num_boost_
round=2000, valid_sets=[dtrain,dval], early_stopping_
rounds=125, verbose_eval=30, )
predicts.append(bst.predict(np.concatenate(
([gdf['X_ts'],gdf['Z_ts']]),-1),num_iteration=bst.
best_iteration)[:,np.newaxis])
predicts = np.concatenate(predicts,-1)
df_slots[slot].add_Base_forecasts(predicts,
forecaster='lightgbm_direct')

```

Once base forecasts are generated in each metadata slot, we can concatenate them into one larger metadata object for meta-learning with the function ‘`meta_concat()`’.

```

from tsfmeta.data import Md_utils
meta_train = Md_utils.meta_concat(df_slots)

```

The metadata class also provides a data loader function to train predefined or custom-designed neural networks for meta-learning.

```

train_dataloader = meta_train.to_
dataloader(train=True, batch_size= 1024)

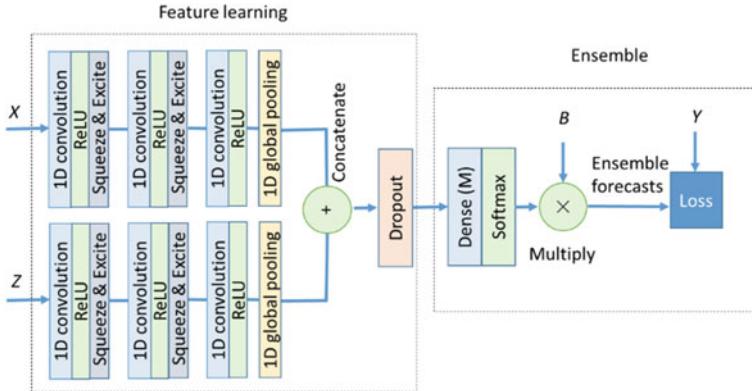
```

### *Meta-Learner*

‘tsfmeta’ predefined four meta-learning neural networks proposed by Ma and Fildes (2021). All the networks take the metadata object as the input.

#### *The Meta-Combination Learner with Encapsulated Features Extractor (‘MetaComb’)*

The network structure of ‘MetaComb’ is shown in Fig. 9.2, which encapsulates temporal convolutional blocks for feature extraction. The network can have at most two channels. One channel simultaneously takes the time series X as the input and the other inputs Z (optional). Both convolutional channels can contain  $k$  (user-defined) stacked temporal convolutional blocks. Each convolutional block contains a convolutional layer and a ReLU activation ( $ReLU(u) = \max(0, u)$ ). The first two convolutional blocks conclude with a squeeze and excite layer (Hu, Shen, Albanie, Sun, & Wu, 2019). The squeeze operation exploits the contextual information outside each filter’s focused feature of the input time series by using a global average pool to generate summary statistics over the learned feature map. The last temporal convolutional block is followed



**Fig. 9.2** Network structure of the ‘MetaComb’

by a global average pooling layer, which reduces the number of parameters in the network. The outputs of the global pooling layer from both channels are concatenated and then fed into a dropout layer to mitigate overfitting. A dense layer with a softmax activation transforms the learned features into weights whose dimension equals the number of base forecasters. Using the weights obtained from the dense layer with softmax activation, the base forecasts  $B$  are weighted and summed to generate the ensemble forecasts.

#### *The Meta-Combination Learner with Unsupervised Hand-Selected Features ('MetaCombFeat')*

The network ‘MetaCombFeat’ takes the judgmentally selected features as the input and uses multiple layers of a fully connected neural network and ReLU activations to process the inputs. Then, a dropout layer is applied to mitigate overfitting. The remaining features were then fed into a dense layer with softmax activation, transforming them into weights. Its structure is shown in Fig. 9.3.

#### *The Model Selection Learner Targets Identifying the Best Base Forecaster ('MetaSelection')*

This meta-learner also uses temporal convolutional blocks to extract features from metadata  $X$  and  $Z$  (optional), and feeds these learned features into a softmax layer. However, the outputs from the softmax

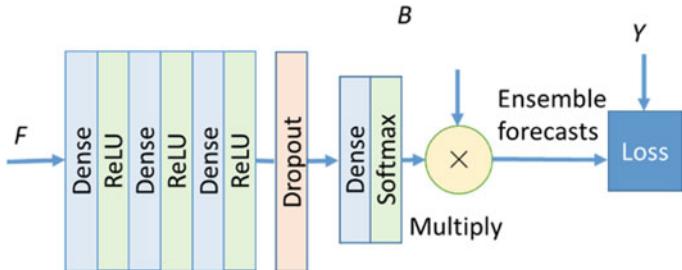


Fig. 9.3 Network structure of the ‘MetaCombFeat’

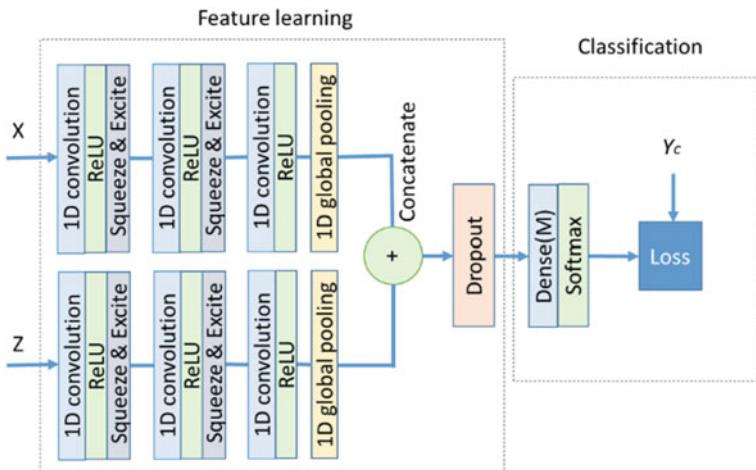


Fig. 9.4 Network structure of the ‘MetaSelection’

are interpreted as the probabilities of which base forecasters provide the most accurate forecasts for the current sales series (Fig. 9.4). During the meta-forecasting phase, the meta-learner combines base forecasts using the probability of being the best as the ensemble weights. It is worth mentioning that the original version of the ‘MetaSelection’ in Ma and Fildes (2021) used the base forecast with the maximum probability of being the best on a time series as the meta-forecast. We here refine this meta-leaner as an ensemble forecaster to improve its forecasting performance.

### The Metaloss Prediction Learner ('MetaLoss')

This meta-learner aims to predict the forecasting errors arising from each base forecaster, which also uses temporal convolutional blocks to extract features from metadata  $X$  and  $Z$  (optional). The structure of this meta-learner is illustrated in Fig. 9.5. During the meta-forecasting phase, the trained meta-learner generates loss predictions for each base forecaster. Then, the loss predictions are used to generate weights to combine the base forecasts.

We also defined a class named 'meta\_learning\_run' as a framework for the training and evaluation of meta-learners. To instantiate the 'meta\_learning\_run' object, we need to provide multiple parameters:

- Meta-learner - Network to be trained and evaluated.
- Loss - Loss criterion.
- Optimizer – The optimization algorithms for network training.
- train\_dataloader - Dataloader streaming training data.

Once we construct the 'meta\_learning\_run' object, we can start the training and evaluation loop via the launch method.

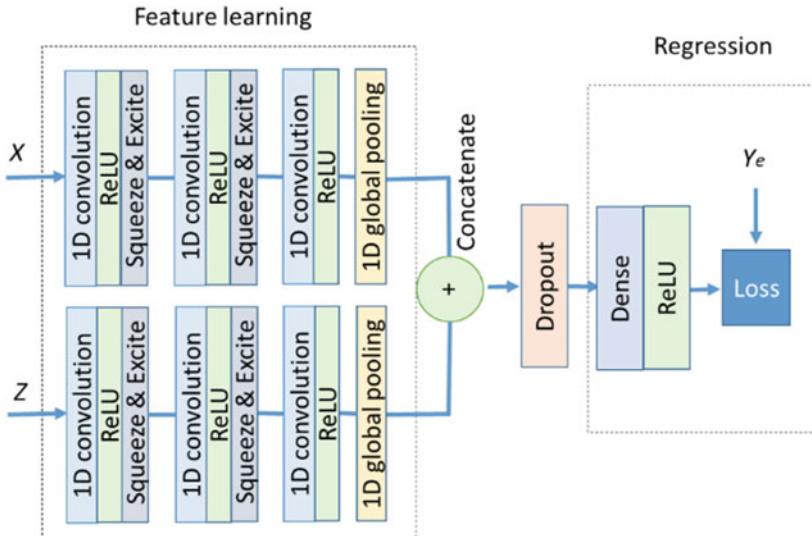


Fig. 9.5 Network structure of 'MetaLoss'

```

from tsfmeta import MetaNN
from tsfmeta.experiments import meta_learning_run
learning_rate = 1e-5
batch_size = 1024
epochs = 10
n_features = meta_train.Z.shape[1]
_, n_forecasters, horizon = meta_train.B.shape
window_x = meta_train.X.shape[2]
model = metalearner.MetaComb(n_features = n_
features , n_forecasters = forecasters ,window_x =
window_x , horizon = horizon)
loss_fn = torch.nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters() , lr=
learning_rate)
meta_learning = meta_learning_run(model, loss_fn,
optimizer, train_dataloader, epochs)

```

Given the trained meta-learner, we can generate meta-forecasts on test metadata.

```

test_dataloader = meta_test.to_dataloader(train=False,
batch_size=batch_size, num_workers=0)
loss, prediction = meta_learning.forecast(test_
dataloader)

```

Lastly, we can evaluate the meta-forecasts compared to the base forecasts using the metadata class function ‘evaluate’, given the metrics we defined in the utility class.

```

from tsfmeta import utils
meta_test.evaluate(prediction.numpy() ,utils.metrics))

```

	<b>ET<sub>S</sub></b>	<b>ADL-1</b>	<b>ARX-1</b>	<b>ELM-1</b>	<b>SVM-1</b>	<b>ADLP-3</b>	<b>RF-7</b>	<b>GBRT-7</b>	<b>ELM-7</b>	<b>MetaComb</b>
MAE	8.638	6.657	6.925	7.580	6.975	6.629	7.094	6.632	6.952	6.338
MSE	635.621	295.741	333.745	448.439	402.104	370.154	432.335	387.955	413.594	267.447
MAPE	0.202	0.178	0.182	0.194	0.181	0.173	0.178	0.173	0.176	0.171
RMSSE	0.790	0.708	0.726	0.767	0.718	0.692	0.719	0.688	0.711	0.676

In the above outputs, the meta-forecasts of ‘Metacomb’ is compared with base forecasts of nine base forecasters, including ExponenTial Smoothing (ETS), Autoregressive Distributed Lag (ADL), Autoregressive integrated moving average with external variables (ARIMAX), Support Vector Machine (SVM), Extreme learning machines (ELM), ADL with data Pooling (ADLP), Random Forest (RF), and Gradient Boosting Regression Trees (GBRT). The number after the model name represents the lags used to build the explanatory feature space.

## EXPERIMENTAL EVALUATION

### *Datasets*

The ‘tsfmeta’ library has been tested on two open-source datasets in different application fields<sup>2</sup>.

- M5 competition dataset. The dataset involves the unit sales of 3,049 products classified into 3 product categories (Hobbies, Foods, and Household) and 7 product departments in which the abovementioned categories are disaggregated. The products are sold across ten stores in 3 States (CA, TX, and WI) of the USA. We here forecast the unit sales at the daily product-store level (30490 time series) with a forecasting horizon of 7 days.
- Kdd2022 Wind Power forecasting dataset. The data are sampled every 10 minutes from each wind turbine in a wind farm (which consists of 134 wind turbines). The data also includes the spatial distribution of wind turbines and the dynamic context factors like temporal, weather, and turbine internal status. We here forecast the daily power outputs with a forecasting horizon of up to 7 days for 134 wind turbines.

### *Predictive Performance*

For both datasets, we have used the first 90% of days from the datasets to build the metadatabase which is then used to train meta-learners,

<sup>2</sup> The demonstrative codes are available at <https://github.com/Shawn-nau/tsfmeta/examples>

and the last 10% days of the data is used for evaluating the forecasting performance of the meta-learners. We used meta-learning networks with tree convolutional blocks with 128, 64, and 32 convolutional filters. The networks were trained for 20 epochs with the Adam optimizer which used a learning rate of  $10^{-5}$  to minimize the mean squared error.

Results on two dataset sets are presented in Table 9.1 and Table 9.2, respectively, and bold numbers denote the best-performing model on the dataset. In the IRI dataset, we trained one local and two global forecasting models as base forecasters, including Simple Exponential Smoothing (SES), LightGBM (Ke et al., 2017) using seven days of lags to build explanatory feature space and using direct strategy to generate multi-horizon forecasts, and LightGBM using 14 days of lags to build explanatory feature space and multiple out strategy to generate multi-horizon forecasts (Ma & Fildes, 2020). Table 9.1 shows that the meta-learner ‘MetaComb’ provides the most accurate forecasts measured by Mean Absolute Error (MAE), Mean Squared Error (MSE), and symmetric Mean Absolute Percentage Error(sMAPE) and is inferior to the other two meta-learners in terms of Root Mean Squared Scaled Error (RMSSE) (Makridakis, Spiliotis, & Assimakopoulos, 2020). The meta-learner ‘MetaSelection’ provides the best forecasts in terms of RMSSE.

In the KDD2022 dataset, in addition to the three base forecasting models we used in the M5 data, we also included one local model of Exponential Smoothing (ES) with a damped-trend, and one global model of Random Forest using seven lags. Similar to the results reported in Table 9.1, Table 9.2 shows that the meta-learner ‘MetaComb’ provides the most accurate forecasts under all the measures. The meta-learner ‘MetaSelection’ also provides improved forecasts over all the base forecasters under all the measures.

**Table 9.1** Performance of base forecasters and meta-learners on the M5 dataset

	MAE	MSE	sMAPE	RMSSE
SES	1.012	4.680	0.638	0.725
LightGBM-7-Direct	0.982	3.704	0.628	0.734
LightGBM-14-MIMO	0.983	3.877	0.630	0.721
MetaComb	0.973	3.649	0.627	0.724
MetaSelection	0.978	3.895	0.633	0.715
MetaLoss	0.982	3.852	0.630	0.721

**Table 9.2** Performance of base forecasters and meta-learners on the KDD2022 dataset

	<i>MAE</i>	<i>MSE</i>	<i>sMAPE</i>	<i>RMSSE</i>
SES	281.95	114,926.82	0.405	1.150
ES	292.72	122,164.45	0.412	1.186
LightGBM-7-Direct	179.68	54,584.87	0.259	0.802
LightGBM-14-MIMO	194.94	61,574.37	0.312	0.832
RandomForest-7-MIMO	198.83	61,918.44	0.276	0.852
MetaComb	176.39	51,594.40	0.256	0.772
MetaSelection	177.94	52,018.48	0.259	0.775
MetaLoss	189.23	59,684.30	0.281	0.825

## CONCLUSIONS AND FUTURE DIRECTIONS

Though the idea of meta-learning is not new, as the big data era arrived, it attracted more interest in the machine learning community to help solve algorithm selection and model tuning problems. In this chapter, we reviewed the current development of meta-learning in time series, introduced a general meta-learning framework for forecasting time series, and discussed the key elements of establishing an effective meta-learning system. We introduced a newly developed python library that facilitates the application of meta-learning in time series forecasting. We showed that the meta-learning method is a promising technique in various time series forecasting applications, especially if we have enough historical data to build big metadata libraries.

For time series forecasting researchers, several important problems in meta-learning remain under-researched. (1) There is a lack of in-depth research on rules for selecting base forecasters. So far, the decision still relies heavily on an analyst's experience and preference despite the research evidence suggesting that a poor choice leads to damaging performance. (2) It is worth exploring and developing more effective neural networks for automatic time series feature extracting in the context of meta-learning. So far, only convolutional neural networks have been tested. (3) Meta-learning is still a black-box method for forecasters. It is an interesting direction for designing meta-learners with explainable machine learning methods, from which researchers can extract explainable rules for model selection or model combination. (4) Most existing meta-learning research in forecasting time series has focused on accuracy; future

research could extend the meta-learning framework to solve quantile or density forecasting problems. (5) It would also be valuable to researchers to build a general cross-sectional giant metadatabase, which aims to help naïve users select suitable forecasting models for their custom time series dataset. Finally, it remains an open question how models such as those described here, including ML base forecasters, can be applied in practical organizational settings: just relying on black-box implementations is seldom satisfactory, either in terms of accuracy or from the final user's perspective.

**Acknowledgements** The first author acknowledges the support of the National Natural Science Foundation of China under Grant No. 72072092; Jiangsu Social Science Foundation under Grant No. 22GLB016; and Jiangsu Philosophy and Social Science Research Major Project under Grant No. 2021SJZDA029.

## REFERENCES

- Aiolli, F. (2011). *Transfer learning by kernel meta-learning*. Paper presented at the International Conference on Unsupervised and Transfer Learning workshop.
- Andrychowicz, M., Denil, M., Colmenarejo, S. G., Hoffman, M. W., Pfau, D., Schaul, T., & Freitas, N. d. (2016). *Learning to learn by gradient descent by gradient descent*. Paper presented at the Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain.
- Arinze, B., Kim, S.-L., & Anandarajan, M. (1997). Combining and selecting forecasting models using rule based induction. *Computers & Operations Research*, 24(5), 423–433. [https://doi.org/10.1016/S0305-0548\(96\)00062-7](https://doi.org/10.1016/S0305-0548(96)00062-7)
- Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., ... Gamboa, H. (2020). TSFEL: Time series feature extraction library. *SoftwareX*, 11, 100456. <https://doi.org/10.1016/j.softx.2020.100456>
- Bengio, Y., Bengio, S., & Cloutier, J. (1991, 8–12 July 1991). *Learning a synaptic learning rule*. Paper presented at the IJCNN-91-Seattle International Joint Conference on Neural Networks.
- Bonissone, P. P. (2012). Lazy Meta-Learning: Creating Customized Model Ensembles on Demand. In J. Liu, C. Alippi, B. Bouchon-Meunier, G. W. Greenwood, & H. A. Abbass (Eds.), *Advances in Computational Intelligence: IEEE World Congress on Computational Intelligence, WCCI 2012, Brisbane, Australia, June 10–15, 2012. Plenary/Invited Lectures* (pp. 1–23). Berlin, Heidelberg: Springer Berlin Heidelberg.

- Cerqueira, V., Torgo, L., Pinto, F., & Soares, C. (2017). *Arbitrated ensemble for time series forecasting*. Paper presented at the (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2017. Lecture Notes in Computer Science, Cham.
- Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307, 72–77. <https://doi.org/10.1016/j.neucom.2018.03.067>
- Collopy, F., & Armstrong, J. S. (1992). Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. *Management Science*, 38(10), 1394–1414. <https://doi.org/10.1287/mnsc.38.10.1394>
- Duncan, G. T., Gorr, W. L., & Szczypula, J. (2001). *Forecasting analogous time series*: Springer US.
- Fildes, R., & Petropoulos, F. (2015). Simple versus complex selection rules for forecasting many time series. *Journal of Business Research*, 68(8), 1692–1701. <https://doi.org/10.1016/j.jbusres.2015.03.028>
- Giraud-Carrier, C. G. (2008). *Metalearning - A Tutorial*. Paper presented at the Tutorial at the 7th international conference on machine learning and applications (ICMLA), San Diego, California, USA.
- Ha, D., Dai, A. M., & Le, Q. V. (2017). HyperNetworks. *ArXiv*, *abs/1609.09106*.
- Henderson, T. (2023). Theft: Tools for handling extraction of features from time series.
- Hospedales, T. M., Antoniou, A., Micaelli, P., & Storkey, A. J. (2022). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44, 5149–5169.
- Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2019). Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. <https://doi.org/10.1109/TPAMI.2019.2913372>
- Hyndman, R., Kang, Y., Montero-Manso, P., Talagala, T., Wang, E., Yang, Y., ... Moorman, J. R. (2019). tsfeatures: Time series feature extraction. R package R package v.1.0.1.
- Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1), 167–177. <https://doi.org/10.1016/j.ijforecast.2019.05.008>
- Kück, M., Crone, S. F., & Freitag, M. (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. Paper presented at the 2016 International Joint Conference on Neural Networks (IJCNN).

- Kadlec, P., & Gabrys, B. (2008). Learnt Topology Gating Artificial Neural Networks. *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008*, 2604–2611.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree*. Paper presented at the Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, California, USA.
- Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10), 2006–2016. <https://doi.org/10.1016/j.neucom.2009.09.020>
- Li, K., & Malik, J. (2017). Learning to Optimize. *ArXiv*, *abs/1606.01885*.
- Ma, P., Zhang, Z., Wang, J., Zhang, W., Liu, J., Lu, Q., & Wang, Z. (2021). Review on the Application of Metalearning in Artificial Intelligence. *Computational Intelligence and Neuroscience*, 2021, 1560972. <https://doi.org/10.1155/2021/1560972>
- Ma, S., & Fildes, R. (2020). Forecasting third-party mobile payments with implications for customer flow prediction. *International Journal of Forecasting*. <https://doi.org/10.1016/j.ijforecast.2019.08.012>
- Ma, S., & Fildes, R. (2021). Retail sales forecasting with meta-learning. *European Journal of Operational Research*, 288(1), 111–128. <https://doi.org/10.1016/j.ejor.2020.05.038>
- Makridakis, S., & Petropoulos, F. (2019). The M4 competition: Conclusions. *International Journal of Forecasting*. <https://doi.org/10.1016/j.ijforecast.2019.05.006>
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M5 Accuracy competition: Results, findings and conclusions.
- Meade, N. (2000). Evidence for the selection of forecasting methods. *Journal of Forecasting*, 19(6), 515–535. [https://doi.org/10.1002/1099-131X\(20001119:6%3c515::AID-FOR754%3e3.0.CO;2-7](https://doi.org/10.1002/1099-131X(20001119:6%3c515::AID-FOR754%3e3.0.CO;2-7)
- Molina, M. D. M., Romero, C., Ventura, S., & Luna, J. M. (2012). *Meta-learning approach for automatic parameter tuning: A case of study with educational datasets*. Paper presented at the 5th International Conference on Educational Data Mining.
- Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2018). *FFORMA: Feature-based Forecast Model Averaging*. Working Paper 19/18.
- O'Hara-Wild, M., Hyndman, R., & Wang, E. (2022). Feasts: Feature extraction and statistics for time series. <https://github.com/tidyverts/feasts/>
- Prudêncio, R. B. C., & Ludermir, T. B. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61(1), 121–137.
- Ravi, S., & Larochelle, H. (2017). *Optimization as a model for few-shot learning*. Paper presented at the International Conference on Learning Representations.

- Reif, M., Shafait, F., & Dengel, A. (2012). Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning*, 87(3), 357–380. <https://doi.org/10.1007/s10994-012-5286-7>
- Rice, J. R. (1976). The Algorithm Selection Problem\*\*This work was partially supported by the National Science Foundation through Grant GP-32940X. This chapter was presented as the George E. Forsythe Memorial Lecture at the Computer Science Conference, February 19, 1975, Washington, D. C. In M. Rubinoff & M. C. Yovits (Eds.), *Advances in Computers* (Vol. 15, pp. 65–118): Elsevier.
- Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1), 131–139. <https://doi.org/10.1162/neco.1992.4.1.131>
- Shah, C. (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting*, 13(4), 489–500. [https://doi.org/10.1016/S0169-2070\(97\)00031-9](https://doi.org/10.1016/S0169-2070(97)00031-9)
- Talagala, T., Hyndman, R., & Athanasopoulos, G. (2018). *Meta-learning how to forecast time series*. <https://EconPapers.repec.org/RePEc:msh:ebswps:2018-6>
- Talagala, T. S., Li, F., & Kang, Y. (2022). FFOMPP: Feature-based forecast model performance prediction. *International Journal of Forecasting*, 38(3), 920–943. <https://doi.org/10.1016/j.ijforecast.2021.07.002>
- Wang, J. X. (2021). Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences*, 38, 90–95. <https://doi.org/10.1016/j.cobeha.2021.01.002>
- Wang, X., Smith-Miles, K., & Hyndman, R. (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, 72(10), 2581–2594. <https://doi.org/10.1016/j.neucom.2008.10.017>
- Widodo, A., & Budi, I. (2013). *Model selection using dimensionality reduction of time series characteristics*. Paper presented at the International Symposium on Forecasting, Seoul, South Korea.
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82. <https://doi.org/10.1109/4235.585893>



## CHAPTER 10

---

# Forecasting Large Collections of Time Series: Feature-Based Methods

*Li Li* , *Feng Li* , and *Yanfei Kang*

## INTRODUCTION

When the vast collection of time series is a common case in economic and other forecasting domains, AI empowers rapid decision-making. The No-Free-Lunch theorem (Wolpert & Macready, 1997) says that no single model always performs the best for all time series. When forecasting large

---

L. Li

School of Economics and Management, University of Science & Technology  
Beijing, Beijing, China  
e-mail: [li.li@ustb.edu.cn](mailto:li.li@ustb.edu.cn)

F. Li

School of Statistics and Mathematics, Central University of Finance and  
Economics, Beijing, China  
e-mail: [feng.li@cufe.edu.cn](mailto:feng.li@cufe.edu.cn)

Y. Kang

School of Economics and Management, Beihang University, Beijing, China  
e-mail: [yanfeikang@buaa.edu.cn](mailto:yanfeikang@buaa.edu.cn)

collections of time series, instead of choosing one model for all the data, features can be used to obtain the most appropriate model or the optimal combination of candidate models per series. With the advancement of AI, more recent literature uses meta-learning to describe the process of automatically acquiring knowledge for forecast model selection/combination, which can be summarized in a general framework in Fig. 10.1. Specifically, the framework is divided into two phases. In the model-training phase, we need a set of reference data that splits into training and testing periods. Then a meta-learner is used to learn the relationship between individual forecasts from the pool, and the forecasting performance is measured by an error metric. Finally, a forecast selection/combination algorithm is trained by minimizing the total forecasting loss. Once the model has been trained, the best forecast method or the combination weights can be decided for any target series given the features.

The idea of meta-learning could date back to the 1970s, when Rice (1976) proposed a groundbreaking framework for algorithm selection problems. Collopy and Armstrong (1992) developed 99 rules based on

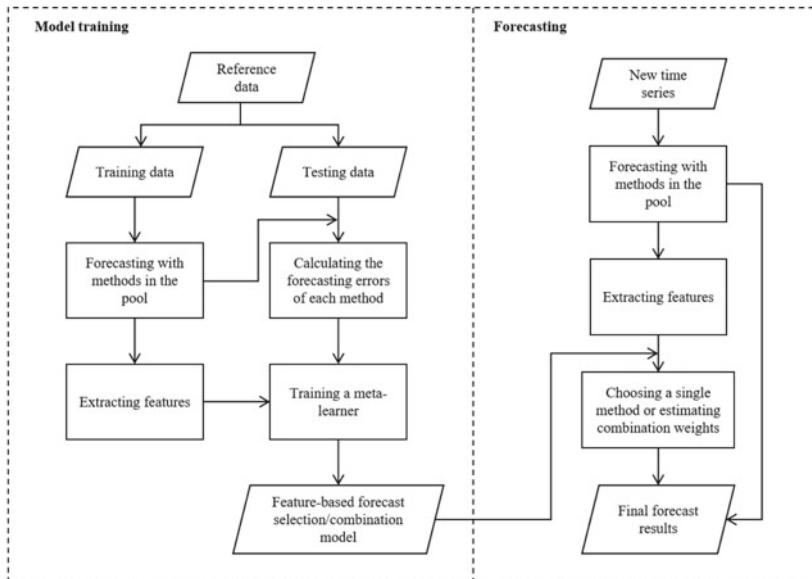


Fig. 10.1 The framework of feature-based methods for forecasting

18 features. Shah (1997) used several features to classify time series and applied discriminant analysis to predict the best-performing model. The term “meta-learning” emerged with machine-learning literature. Prudêncio and Ludermir (2004) presented an original work that used “meta-learning” in model selection for time series forecasting. They investigated two meta-learning approaches based on two case studies. Wang et al. (2009) focused on rule induction for selecting a forecasting method by understanding the nature of historical data. Lemke and Gabrys (2010) compared different meta-learning approaches to investigate which model worked best in which situation. More recently (Kück et al., 2016) proposed a meta-learning framework based on neural networks for forecast selection. Further evidence in favor of meta-learning is given in Talagala et al. (2022, 2023).

Using meta-learning to obtain weights for forecast combinations has received increased attention lately. The feature-based forecast model averaging (FFORMA) framework proposed by Montero-Manso et al. (2020) employed 42 features to estimate the optimal combination weights of nine forecasting methods based on extreme gradient boosting (XGBoost, Chen and Guestrin [2016]). Kang et al. (2020) used 26 features to predict the performances of nine forecasting methods with synthetic time series data, and obtained the combination weights by a tailored softmax function of the predicted forecasting errors. Li et al. (2020), Kang et al. (2022), Wang et al. (2022b), and Talagala et al. (2022) extended the framework of FFORMA based on new types of features and a variety of meta-learners and real-world applications. The representative research in feature-based forecasting is summarized in Table 10.1.

In practice, a typical feature-based forecasting method requires the forecaster to make the following decisions: (1) What training data to use to train the meta-learner? (2) What features to use to describe the time series characteristics? (3) What forecasts to include to formulate the forecast pool? (4) How to use meta-learners in applications? As such, around the above questions, the remainder of this chapter includes the following sections.

- **Data generation.** The performance of any time series mining algorithm, including forecasting, depends on the diversity of the training data. In the era of big data, many industries have ample and diverse historical data that can be used to train forecasting models. However,

**Table 10.1** Overview of literature on feature-based forecasting

<i>Authors (year)</i>	<i>Type</i>	<i>Meta-learning algorithm</i>	<i>Number of features used</i>
Collopy and Armstrong (1992)	Combination	Rule-based induction	18
Shah (1997)	Selection	Discriminant analysis	26
Prudêncio and Ludermir (2004)	Selection	C4.5 decision tree, Neural network classifier	14, 16
Wang et al. (2009)	Selection	C4.5 decision tree, SOM clustering	9
Lemke and Gabrys (2010)	Selection	Feed-forward neural network, decision tree, support vector machine	24
Petroopoulos et al. (2014)	Selection	Neural network	9
Kück et al. (2016)	Selection	Neural network	127
Talagala et al. (2023)	Selection	Random forest	33
Kang et al. (2020)	Combination	Nonlinear regression	26
Montero-Manso et al. (2020)	Combination	XGBoost	42
Kang et al. (2022)	Combination	XGBoost	Depend on the number of forecasting models
Li et al. (2020)	Combination	XGBoost	Depend on time series imaging
Li et al. (2022b)	Combination	XGBoost	9
Wang et al. (2022b)	Combination	Generalized additive models (GAMs)	43
Talagala et al. (2022)	Selection	Bayesian multivariate surface regression	37

there are still cases such as the lack of historical data, or it would be very difficult or expensive to obtain data. To solve these problems, we introduce some methods for time series generation.

- **Feature extraction.** This section introduces a series of time series features widely used in forecasting and ways to automatically extract time series features without the need for expert knowledge and human interaction. The methods for the automation of feature extraction include time series imaging, calculation of forecast diversity, and automatic feature selection from thousands of features.
- **Forecast trimming.** This section discusses methods to decide which forecasts should be included especially for combinations. The gains from forecast combinations highly depend on the quality of the pool of forecasts to be combined and the estimation of combination weights. There are three main factors to influence the quality of the forecasting pool, which are accuracy, robustness (measured by the variance of the accuracy), and diversity (independent information contained in the component forecasts) of the individual forecasts. We review some classical methods and provide a way to address the accuracy, robustness, and diversity simultaneously.
- **Some practical forecasting issues.** This section discusses some practical issues that arise in feature-based forecasting. For example, intermittent demand with several periods of zero demand is ubiquitous in practice. In addition, uncertainty estimation can indicate risk information, which is useful in actual decisions. We provide some feature-based solutions for the two aforementioned issues. The choices of meta-learners, as well as loss functions, are also discussed.

## DATA GENERATION

The performance of any time series forecasting method depends on the diversity of the training data, so that the evaluation of the method can be generalized to a wide range of future data. Therefore, there is a need for comprehensive time series benchmarks. Although some attempts have been made in certain time series tasks, such as the M4 dataset for a recent time series forecasting competition Makridakis et al. (2020), the time series area lacks diverse and controllable benchmarking data for forecasting evaluation. This section reviews some recent work for time series generation. Section “[Diverse Time Series Generation](#)” introduces a series

of methods to generate diverse time series and mainly focuses on Gaussian mixture autoregressive (MAR) models. Section “[Time Series Generation with Target Features](#)” reviews some advanced methods that can generate time series with controllable characteristics.

### *Diverse Time Series Generation*

In recent years, research has shown great potential to use generated data for algorithm learning under certain application domains. Such examples can be found in evaluating statistical methods for temporal outbreak detection (Bédubourg & Le Strat, [2017](#); Lotze & Shmueli, [2009](#)), examining neural network forecasting (Zhang et al., [2001](#)), and so on. In previous studies of time series generation, some scholars focused on the shapes of given time series, or on some predefined types of time series. Vinod and López-de Lacalle ([2009](#)) used the maximum entropy bootstrap to generate ensembles for time series inference by extending the traditional iid bootstrap to nonstationary and dependent data. Bagnall et al. ([2017](#)) simulated time series from different shape settings by placing one or more shapes on a white noise series. The simulators are designed for different feature spaces to evaluate classification algorithms. However, it is impossible to create simulated time series covering the possible space of all time series, which limits the reproducibility and applicability of the tested methodology.

Kang et al. ([2020](#)) proposed an efficient and general approach of GeneRAting TIme Series (gratis) with diverse and controllable characteristics, named GRATIS, based on MAR models to generate a wide range of non-Gaussian and nonlinear time series. Mixture transition distribution models were first developed by Li et al. ([2010](#)) to capture non-Gaussian and nonlinear features, and generalized to MAR models later (Wong & Li, [2000](#)).

MAR models consist of multiple stationary or nonstationary autoregressive components. A  $K$ -component MAR model can be defined as:

$$F(x_t | \mathbb{F}_{-t}) = \sum_{k=1}^K \alpha_k \Phi\left(\frac{x_t - \phi_{k0} - \phi_{k1}x_{t-1} - \cdots - \phi_{kp_k}x_{t-p_k}}{\sigma_k}\right), \quad (10.1)$$

where  $F(x_t | \mathbb{F}_{-t})$  is the conditional cumulative distribution of  $x_t$  given the past information  $\mathbb{F}_{-t} \subseteq \{x_{t-1}, \dots, x_{t-p_k}\}$ ,  $\Phi(\cdot)$  is the cumulative distribution function of the standard normal distribution,  $x_t - \phi_{k0} - \phi_{k1}x_{t-1} - \dots - \phi_{kp_k}x_{t-p_k}$  is the autoregressive term in each mixing component,  $\sigma_k > 0$  is the standard error,  $\sum_{k=1}^K \alpha_k = 1$  and  $\alpha_k > 0$  for  $k = 1, 2, \dots, K$ . Denoted as MAR( $K; p_1, p_2, \dots, p_k$ ), it is actually finite mixtures of  $K$  Gaussian AR models.

A significant difference in Kang et al.'s (2020) data generation process compared to typical simulation processes used in the literature is that the authors used distributions instead of fixed values for the parameters. This allows for the generation of diverse time series instances. The parameter settings are analogous to noninformative priors in the Bayesian contexts, that is, the diversity of the generated time series should not rely on the parameter settings.

So far, we have focused on time series with only one seasonal pattern. However, many time series exhibit multiple seasonal patterns of different lengths, especially those series observed at a high frequency (such as daily or hourly data). Simulation of multi-seasonal time series involves the weighted aggregation of simulated time series with the corresponding frequencies. A simulated multisessional time series  $x_t$  with  $M$  seasonal patterns can be written as  $x_t = \sum_{m=1}^M \omega_m x_{F_m, t}$ , where  $m = 1, 2, \dots, M$ ,  $x_{F_m, t}$  is the  $m$ th simulated time series with frequency  $F_m$ , and weight  $\omega_m$  satisfies  $\sum_{m=1}^M \omega_m = 1$  and  $0 < \omega_m < 1$ . The weights can be obtained by  $\omega_m = \gamma_m / \sum_{r=1}^M \gamma_r$ , where  $\gamma_m \sim U(0, 1)$ .

The critical merits of MAR models for nonlinear time series modeling are (Li et al., 2010; Wong & Li, 2000): (a) MAR models can capture extensive time series features in principle based on a sufficiently diverse parameters space and a finite number of components, (b) mixtures of stationary and nonstationary components can yield both stationary and nonstationary process with MAR, (c) the conditional distributions of time series change with time, which allows for evolution with historical information, (d) MAR models can handle complicated univariate and multivariate time series with different values of frequencies and seasonality, and (e) MAR models can also capture features such as multimodality, heavy tails, and heteroskedasticity.

### *Time Series Generation with Target Features*

In time series analysis, researchers with a particular focus may be only interested in a specific area of the feature space or a subset of features, for example, heteroskedasticity and volatility in financial time series, trend and entropy in microeconomic time series, or peaks and spikes in energy time series. Practitioners may also want to mimic more time series from their limited collection of real data, such as sales records for new products and health metrics for rare diseases. Therefore, the efficient generation of time series with target features of interest is another crucial problem to address.

Kang et al. (2017) used a genetic algorithm (GA) to generate new time series to fill in any gaps in a two-dimensional instance space. Kegel et al. (2017) applied STL method to estimate the trend and seasonal component of a time series, which are modified using multiplicative factors to generate new time series. Kang et al.’s (2017) evolutionary algorithm is quite general, but computationally slow, whereas the STL-based algorithm from Kegel et al. (2017) is much faster but can only generate series that are additive in trend and seasonality. Talagala et al. (2023) augmented the set of observed time series by simulating new time series similar to those to form a larger dataset for training the model-selection classifier. The simulated series rely on the assumed data generating processes (DGPs), which are exponential smoothing models and ARIMA models.

Kang et al. (2020) used a GA to tune the MAR model parameters until the distance between the target feature vector and the feature vector of a sample of time series simulated from the MAR is close to zero. The parameters for the MAR model, the underlying DGP, can be represented as a vector  $\Theta = \{\alpha_k, \phi_i\}$ , for  $k = 1, \dots, K$ ,  $i = k_0, \dots, kp_k$  in Eq. (10.1). The authors developed an R package *gratis* for the time series generation which is available from CRAN. The R package *gratis* provides efficient algorithms for generating time series with diverse and controllable characteristics.

## FEATURE EXTRACTION

A feature can be any function computed from a time series. Examples include a simple mean, the parameter of a fitted model, or some statistic intended to highlight an attribute of the data. A unique “best” feature representation of a time series does not exist. What features are used

depends on both the nature of the time series being analyzed and the purpose of the analysis. For example, consider the mean as a simple time series feature. If some time series contain unit roots, then the mean is not a meaningful feature without additional constraints on the initial values of the time series. Even if the series are all stationary, if the purpose of our analysis is to identify the best forecasting method, then the mean is probably of no value. This example shows that it is difficult to formulate general desirable properties of features without knowledge of both the time series properties and the required analysis. Section “[Time Series Features](#)” reviews a series of features that have been found helpful in time series exploration.

Recently, a series of automated approaches to extract time series features have been put forward, which do not need experts to select features manually and require limited human interaction. Section “[Time Series Imaging](#)” introduced a new tool of time series imaging proposed by Li et al. (2020). Li et al. (2020) transformed time series into recurrence plots, from which local features can be extracted using computer vision algorithms. The study reviewed above depends on using each time series’s historical, observed data. Estimating the features might not be feasible or robust in the case of a limited number of available past observations. Kang et al. (2022) proposed to focus on the produced forecasts to extract features, which is discussed in section “[Forecast Diversity](#)”. When the chosen features involve large numbers, estimating them might increase the computational time required. Theodorou et al. (2022) provided a methodological approach to select from thousands of features, which is discussed in section “[Automatic Feature Selection](#)”.

### *Time Series Features*

We use statistical functions to compute the features of that time series, such as the mean, minimum or maximum. For example, all the autocorrelation values of a series can be considered features of that series. We can also summarize the autocorrelation values to produce new features. The sum of the first ten squared autocorrelation coefficients is a useful summary of how much autocorrelation there is in a series. We can also compute the autocorrelation values of the changes in the series between periods. One could create a new time series consisting of the differences between consecutive observations. Occasionally it is useful to apply

the same differencing operation again, so one could compute the differences of the differences. Another related approach is to compute seasonal differences of a series. The autocorrelations of these differenced series may provide useful information. In addition, the STL (a seasonal-trend decomposition) is the basis for several features. A time series decomposition can be used to measure the strength of trend and seasonality in a time series.

Based on the literature review, time series features vary from tens to thousands, and choosing different sets of features will inevitably result in different forecasts and varied performance. Therefore, choosing and weighing time series features is a common challenge in feature-based forecasting. Several packages, including functions for computing features and statistics from time series, have been developed and widely used in the forecasting community. For example, the **tsfeatures** (Hyndman et al., 2020) and **feasts** (O'Hara-Wild et al., 2022) packages for R include statistics that are computed on the first- and second-order differences of the raw series, account for seasonality, and exploit the outputs of popular time series decomposition methods, among others; the **tsfresh** (Christ et al., 2018) package for Python involves thousands of features, such as basic time series statistics, correlation measures, entropy estimations, and coefficients of standard time series forecasting and analysis methods; the **catch22** (Lubba et al., 2019) includes 22 time series features selected from over 7000 features in **htsfa** (Fulcher et al., 2013). Several feature-based forecast combination models used the features in **tsfeatures** as the basic pool of features (Kang et al., 2020; Li et al., 2022a; Montero-Manso et al., 2020; Talagala et al., 2022; Wang et al., 2022b).

### *Automation of Feature Extraction*

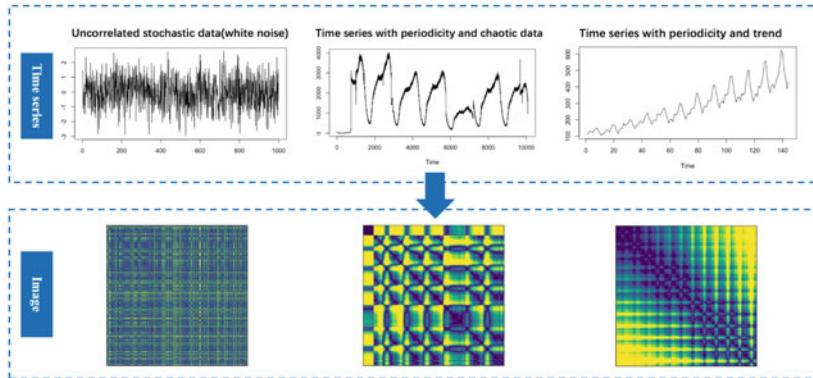
#### *Time Series Imaging*

Li et al. (2020) introduced an automated approach to extract time series features based on a time series imaging process. In the first step, Li et al. (2020) encoded the time series into images using recurrence plots. In the second step, image processing techniques extract time series features from images. Li et al. (2020) considered two different image feature extraction approaches: a spatial bag-of-features (SBoF) model and convolutional neural networks (CNNs). Recurrence plots (RPs) provide a way to visualize the periodic nature of a trajectory through a phase space and can contain all relevant dynamical information in the time series. Figure 10.2

shows three typical recurrence plots. They reveal different patterns of recurrence plots for time series with randomness, periodicity, chaos, and trend.

The SBoF model used in Li et al. (2020) for time series feature extraction consists of three steps: (i) detect key points with the scale-invariant feature transform (SIFT) algorithm and find basic descriptors with k-means; (ii) generate the representation based on the locality-constrained linear coding (LLC) method; and (iii) extract spatial information by spatial pyramid matching (SPM) and pooling. For the details in each step, see Li et al. (2020).

An alternative to SBoF for image feature extraction is to use a deep CNN, which has achieved great breakthroughs in image processing. In this task, the deep network is trained on the large ImageNet dataset, and the pre-trained network is publicly available. For the CNN model, the closer the layer is to the first layer, the more general features can be extracted; the closer the layer is to the back layer, the more specific features for classification tasks can be extracted. To extract time series features, the parameters of all the previous layers are fixed except for the last fully connected layer, and the last layer is fine-tuned. With the trained model, the final representation of the time series can be obtained. The



**Fig. 10.2** Typical examples of recurrence plots (top row) for time series data with different patterns (bottom row): uncorrelated stochastic data, i.e., white noise (left), a time series with periodicity and chaos (middle), and a time series with periodicity and trend (right)

code to implement the above algorithm is available at <https://github.com/lixixibj/forecasting-with-time-series-imaging>.

### *Forecast Diversity*

The task of selecting appropriate features to build forecasting models is often challenging. Even if there was an acceptable way to define the features, existing features are estimated based on historical patterns, which are likely to change in the future. Other times, the estimation of the features is infeasible due to limited historical data. To solve these problems, Kang et al. (2022) suggested a change of focus from the historical data to the produced forecasts to extract features. Kang et al. (2022) used out-of-sample forecasts to obtain weights for forecast combinations by amplifying the diversity of combined methods.

For a given time series  $\{y_t, t = 1, 2, \dots, T\}$ , the  $h$ th step forecast produced by the  $i$ th individual method is denoted as  $f_{ih}$ , where  $i = 1, 2, \dots, M$  and  $h = 1, 2, \dots, H$ . Furthermore,  $M$  and  $H$  are the number of methods in the forecast pools and the forecast horizon, respectively. Let  $f_{ch}$  be the  $h$ th step combined forecast given by  $\sum_{i=1}^M w_i f_{ih}$ , where  $w_i$  is the combination weight for the  $i$ th method. The overall mean squared error of a weighted forecast combination model  $\text{MSE}_{\text{comb}}$  over the whole forecast horizon  $H$  can be written as follows.

$$\begin{aligned}
 \text{MSE}_{\text{comb}} &= \frac{1}{H} \sum_{i=1}^H \left( \sum_{i=1}^M w_i f_{ih} - y_{T+h} \right)^2 \\
 &= \frac{1}{H} \sum_{i=1}^H \left[ \sum_{i=1}^M w_i (f_{ih} - y_{T+h})^2 - \sum_{i=1}^M w_i (f_{ih} - f_{ch})^2 \right] \\
 &= \frac{1}{H} \sum_{i=1}^H \left[ \sum_{i=1}^M w_i (f_{ih} - y_{T+h})^2 - \sum_{i=1}^{M-1} \sum_{j>i}^M w_i w_j (f_{ih} - f_{jh})^2 \right] \\
 &= \sum_{i=1}^M w_i \text{MSE}_i - \sum_{i=1}^{M-1} \sum_{j>i}^M w_i w_j \text{Div}_{i,j},
 \end{aligned} \tag{10.2}$$

where  $\text{MSE}_i$  represents the mean squared error for the  $i$ th method.  $\text{Div}_{i,j}$  denotes the degree of diversity between the  $i$ th and  $j$ th method in the forecast method pool.

Equation (10.2) says that the mean squared error of the combined forecast is guaranteed to be less than or equal to the weighted mean squared error of the individual forecasts. The second term in the last line of Eq. (10.2) tells us how diverse the individual forecasts are. The more diversity existing in the forecast method pool leads to overall better forecasting accuracy.

Kang et al. (2022) proposed to use the pairwise diversity measures as a set of features for each time series. To make the diversity comparable between time series with different scales, the scaled diversity can be defined as:

$$s\text{Div}_{i,j} = \frac{\sum_{h=1}^H (f_{ih} - f_{jh})^2}{\sum_{i=1}^{M-1} \sum_{j=i+1}^M \left[ \sum_{h=1}^H (f_{ih} - f_{jh})^2 \right]}. \quad (10.3)$$

Given a time series dataset with  $N$  time series and a forecasting pool containing  $M$  methods,  $H$ -step point forecasts are produced by the methods. Then we can get an  $M \times H$  matrix for each time series to forecast. Based on Eq. (10.3), the pairwise forecast diversity among the  $M$  methods can be calculated. Thus, for each time series, we get an  $M \times M$  symmetric matrix, which can be transformed into a feature vector with  $M(M - 1)/2$  pairwise diversity measures.

The merits of using diversity for forecast combinations are twofold. First, the process of extracting diversity is straightforward and interpretable. The algorithm for measuring the diversity between different methods involves a simple calculation, and hence, it can reduce the computational complexity when extracting features. Secondly, although traditional feature extraction methods usually depend on the manual choice of an appropriate set of features, our approach can be applied automatically without needing expert knowledge and human interaction.

### *Automatic Feature Selection*

A large number of features significantly increase the dimensionality and complexity. Therefore, a feature selection procedure is necessary. Theodorou et al. (2022) used three steps to capture the key features of the M5 competition data, including statistical pre-filtering, performance evaluation, and redundancy minimization.

The statistical pre-filtering step aims to remove the non-significant features rather than choosing the most meaningful ones. It involves the z-score standardization of the features and eliminating those that display the

same or similar values across different series. The features that can effectively differentiate distinct classes in the dataset statistically significantly are selected.

To evaluate the quality of the extracted features, Theodorou et al. (2022) employed the Regressional ReliefF (RReliefF) algorithm (Robnik-Sikonja & Kononenko, 2003), which is an extension of the ReliefF algorithm for regression problems. ReliefF is a robust filtering method used to select features in multi-class classification problems, with the basic idea of identifying feature differences between nearest instance pairs. Specifically, ReliefF calculates a score for each feature and performs feature selection accordingly. Li et al. (2022a) constructed a multi-class classification problem by labeling the model that performed the best and employed the ReliefF algorithm to rank the features. Then some of the top features are selected for Li et al.'s (2022a) Bayesian forecast combination framework. An R package **febama** is developed for this purpose at <https://github.com/lily940703/febama>.

Theodorou et al. (2022) employed hierarchical clustering to reduce the redundancy in the obtained top-performing features using the Pearson correlation distance (cosine distance) with complete linkage at a threshold of 0.2. The process makes the pairwise correlation coefficients of the features in the same cluster larger than 0.8, thus forming clusters of similarly performing features. The vectors generated from the previous step (performance evaluation) are the input to the clustering method. In each cluster, the feature with the largest mean quality score is selected.

After completing the above feature selection procedure, Theodorou et al. (2022) obtained a set of 42 features, including information about the coefficients of the discrete Fourier transform, the variance and distribution of the data, the entropy and linear trend of the series, the statistics of popular tests for time series analysis, and so on. Although some of the selected features are challenging to interpret, the method of automatic feature selection is more flexible and generic compared with alternative methods that arbitrarily use a limited set of features.

## FORECAST TRIMMING FOR COMBINATION

Forecast combination is widely used as a preferred strategy over forecast selection due to its ability to reduce the uncertainty of identifying a single “best” forecast. Nonetheless, sophisticated combinations are often empirically defeated by simple averaging, which is commonly attributed to the

weight estimation error. The issue becomes more serious when dealing with a forecast pool containing a large number of individual forecasts. Therefore, forecast trimming algorithms are indispensable to identify an optimal subset from the original forecast pool for forecast combination tasks.

When determining which forecasts should be combined, it is crucial to look at the characteristics of the available alternative forecasts, among which robustness, accuracy, and diversity are the most frequently emphasized in the literature (Atiya, 2020; Budescu & Chen, 2015; Lichtendahl & Winkler, 2020; Thomson et al., 2019). Section “[Accuracy, Robustness and Diversity](#)” discusses the three factors and reviews some classical methods for forecast trimming. Wang et al. (2022a) took into account the robustness, accuracy, and diversity of the forecast pool simultaneously and proposed a novel algorithm, which is discussed in section “[Forecast Trimming](#)”.

### *Accuracy, Robustness, and Diversity*

**Accuracy.** Forecast combinations base their performance on the mean level of the accuracy of the individual forecasts to be combined. Including a poorly performing forecast in a combination is likely to decrease the accuracy of the combined forecast. Therefore, it makes intuitive sense to eliminate the worst performers from the forecast pool based on some performance criteria. Kourentzes et al. (2019) proposed a heuristic called “forecast islands” to automatically formulate forecast pools and found it beneficial to the accuracy of the combined forecasts. The heuristic proposed by Kourentzes et al. (2019) uses top  $q$  quantiles for forecast combinations. The cut-off point of how many quantiles is determined automatically rather than arbitrarily. Discarding a set of worst performers is also the most common strategy in the literature on the “wisdom of crowds”, which is referred to as “select-crowd” strategy (see, e.g., Budescu & Chen, 2015; Goldstein et al., 2014; Mannes et al., 2014).

**Robustness.** Lichtendahl and Winkler (2020) highlighted the importance of robustness in dealing with forecast combination problems. The characteristics of a time series generally change over time, so the pattern detected by a forecasting model in the training set may not continue in the validation and test sets. When dealing with a set of time series, the robustness of a given individual forecast can be assessed using the variance of its accuracy across different series. Lichtendahl and

Winkler (2020) suggested balancing the trade-offs between accuracy and robustness when identifying a subset from the available forecast pool.

**Diversity.** The performance of forecast combinations also relies on the independent information contained in the individual forecasts, which relates to the diversity of the forecast pool. Mannes et al. (2014) and Thomson et al. (2019) emphasized that diversity and accuracy are the two crucial factors in manipulating the quality of the forecast combinations. The benefits of diversity are proved theoretically by Atiya (2020), who decomposed the mean squared error (MSE) into a bias term and a variance term. Atiya (2020) found that the decrease in the variance of the forecast combination becomes larger as the correlation coefficients among the individual forecasts decrease. In the field of forecast combinations, some scholars have researched to use diversity measures as additional inputs to improve forecast combinations. Kang et al. (2022) proposed a diversity-based forecast combination framework using only diversity features. The definition of diversity in Kang et al. (2022) is discussed in section “[Forecast Diversity](#)”.

Until recently, the diversity of the forecast pool has been considered for forecast trimming. Cang and Yu (2014) designed an optimal subset selection algorithm using mutual information, which measures dependence between individual forecasts. Lichtendahl and Winkler (2020) eliminated individual forecasts with low accuracy and highly correlated errors. However, considering accuracy and diversity in isolation is questionable, as a poorly performing forecast in the pool may still benefit forecast combinations by injecting diversity into the pool. To solve the problem, Wang et al. (2022a) proposed a new algorithm for forecast trimming, taking the robustness and accuracy of the individual forecasts into account, as well as the degree of diversity of the forecast pool, which is discussed in the next section.

### *Forecast Trimming*

Wang et al. (2022a) used the Mean Squared Error for Coherence (MSEC, Thomson et al. [2019]), also known as Div in Kang et al. (2022), to assess the degree of diversity between individual forecasts. Let  $f_{i,h}$  be the  $h$ th step forecast of the  $i$ th forecaster in a given forecast pool, where  $i = 1, 2, \dots, M$  and  $h = 1, 2, \dots, H$ . The MSEC between the  $i$ th and  $j$ th

methods in the forecast pool is defined as

$$\text{MSEC}_{i,j} = \frac{1}{H} \sum_{h=1}^H (f_{i,h} - f_{j,h})^2. \quad (10.4)$$

A value of zero for this measure indicates that the two individuals ( $i$  and  $j$ ) have made identical forecasts, and a larger value indicates a higher degree of diversity.

Kang et al. (2022) showed that the overall MSE of a weighted combined forecast,  $\text{MSE}_{\text{comb}}$ , can be decomposed into component measures corresponding to accuracy (performance) and diversity (coherence), as follows:

$$\text{MSE}_{\text{comb}} = \sum_{i=1}^M w_i \text{MSE}_i - \sum_{i=1}^{M-1} \sum_{j=2, j>i}^M w_i w_j \text{MSEC}_{i,j}. \quad (10.5)$$

For the process of this decomposition, refer to Eq. (10.2). Inspired by the decomposition, Wang et al. (2022a) proposed a new criterion for forecast trimming to identify an optimal subset for forecast combinations, denoted as Accuracy-Diversity Trade-off (ADT). The ADT criterion is given by

$$\begin{aligned} \text{ADT} &= \text{AvgMSE} - \kappa \text{AvgMSEC} \\ &= \underbrace{\frac{1}{M} \sum_{i=1}^M \text{MSE}_i}_{\text{mean level of accuracy}} - \kappa \underbrace{\frac{1}{M^2} \sum_{i=1}^{M-1} \sum_{j=2, j>i}^M \text{MSEC}_{i,j}}_{\text{overall diversity}}, \end{aligned} \quad (10.6)$$

where  $\kappa$  is a scale factor and  $\kappa \in [0, 1]$ .

Then a new trimming algorithm based on ADT is proposed, denoted as RAD, which simultaneously addresses robustness, accuracy, and diversity. To apply RAD algorithm, the available in-sample data needs to be divided into the training set and the validation set. The training set is used to fit statistical or machine-learning models. The forecasts from these fitted models are then evaluated against the validation set, whose length is the same as the required out-of-sample horizon, and then used to identify an optimal subset based on the RAD algorithm.

Firstly, considering an initial forecaster set  $\mathbb{S}$ , we can apply Tukey's fences approach to exclude the individuals that lack robustness. Specifically, the individual forecasters with the variance of absolute errors

exceeding  $Q_3 + 1.5(Q_3 - Q_1)$  are removed, where  $Q_1$  and  $Q_3$  are the first and third quantiles of the respective values across individual forecasters from the set  $\mathbb{S}$ . Then for individual forecaster  $i$ , calculate the ADT value of the remaining set after removing  $i$  from  $\mathbb{S}$ , find the minimum ADT value and exclude the corresponding forecaster. Repeat the above step until there is a non-significant reduction of the ADT value for  $\mathbb{S}$  or until  $\mathbb{S}$  contains only two forecasters. The optimal subset identified by RAD approach has been proved to achieve good performance and robustness for both point forecasts and prediction intervals (Wang et al., 2022a).

## SOME PRACTICAL FORECASTING ISSUES

As mentioned above, some scholars used meta-learners to construct feature-based forecast selection/combination methods in recent years, such as using random forecast (Talagala et al., 2023), XGBoost (Kang et al., 2022; Montero-Manso et al., 2020), GAMs Wang et al. (2022b). Montero-Manso et al. (2020) proposed a feature-based framework for forecast combination named FFOMA, which used meta-learning to model the links between time series features and the out-of-sample performance of individual forecasting models. The following optimization problem is solved to obtain the combination weights:

$$\arg \min_w \sum_{n=1}^N \sum_{i=1}^M w(F_n)_i \times L_{ni}, \quad (10.7)$$

where  $F_n$  is the feature vector of the  $n$ th time series,  $N$  is the number of time series, and  $M$  is the number of forecasting methods.  $L_{ni}$  is the forecast loss of the  $i$ th method for the  $n$ th time series.

Montero-Manso et al.'s (2020) method placed second in the M4 competition and inspired a series of expanded studies. Li et al. (2020) introduced an automated approach to extract time series features based on time series imaging, as discussed in section “[Time Series Imaging](#)”. Kang et al. (2022) developed FFOMA by using diversity to replace time series features in Eq. 10.7. It can reduce the computational complexity and compute automatically without the need for a manual choice of an appropriate set of features. See section “[Forecast Diversity](#)” for more details. Montero-Manso et al. (2020), Kang et al. (2022), and Li et al.'s (2020) work are all based on M4 competition dataset, containing 100,000 time series with different seasonal periods

from different domains such as demographics, finance, and industries. In addition, this chapter discusses other applications of the aforementioned feature-based methods in the following complex issues.

Demand forecasting is the basis for most planning and control activities in any organization. In particular, demand may appear sporadically, with no demand in some periods, leading to an intermittent appearance. Intermittent demand items are prevalent in many industries, including the automotive, IT, and electronics sectors. Their inventory implications are dramatic and forecasting their requirements is challenging. Li et al. (2022b) examined the empirical outcomes of some existing forecast combination methods and proposed a generalized feature-based framework for intermittent demand forecasting, which is discussed in section “[Intermittent Demand](#)”.

In recent years, probabilistic forecasts have received increasing attention. For example, the recent M4 and the M5 uncertainty competitions Makridakis et al. (2022) encouraged participants to provide probabilistic forecasts of different types as well as point forecasts. Probabilistic forecasts are appealing for enabling optimal decision-making with a better understanding of uncertainties and risks. Interval forecasts form a special case of probabilistic forecasts, which are evidently easier to implement. Wang et al. (2022b) investigated how time series features affect the performances of prediction intervals from different methods. The proposed feature-based interval forecasting framework is discussed in section “[Uncertainty Estimation](#)”.

### *Intermittent Demand*

What makes intermittent demand challenging to forecast is that there are two sources of uncertainty: the sporadic demand occurrence, and the demand arrival timing. Despite that intermittent demand forecasting has obtained some research achievements in recent decades, limited attention has been given to combination schemes for intermittent demand forecasting. Li et al. (2022b) provided a discussion and comparison of forecast combination methods in the context of intermittent demand forecasting and developed a feature-based combination framework for intermittent demand. The proposed framework has been shown to improve the accuracy of point and quantile forecasts.

Li et al. (2022b) defined a broad pool for intermittent demand forecasting. The pool includes traditional forecasting models and intermittent

demand forecasting methods to ensure diversity in the pool. The authors built an XGBoost model to learn the relationship between features and combination weights. Time series features and the diversity of the pool of methods are all valid inputs for the forecast combination model. Li et al. (2022b) considered nine explainable time series features, which imply the intermittency, volatility, regularity, and obsolescence of intermittent demand, and applied the scaled diversity defined in Kang et al. (2022) for intermittent demand forecasting.

In the literature, some scholars argue that traditional accuracy measures based on absolute errors are unsuitable for intermittent demand, because a flat zero forecast is frequently “best” for these measures when the demand is highly intermittent (Kolassa, 2016). The root mean squared scaled error (RMSSE) is used to measure the performance of point forecasts for intermittent demand in Li et al.’s (2022b) work, which can be obtained as:

$$\text{RMSSE} = \sqrt{\frac{1}{H} \frac{\sum_{h=1}^H (y_{T+h} - \hat{y}_{T+h})^2}{\frac{1}{T-1} \sum_{t=2}^T (y_t - y_{t-1})^2}}, \quad (10.8)$$

where  $H$  is the forecasting horizon.  $\hat{y}_{T+h}$  is the  $h$ th step forecast generated from a series of observed values  $\{y_t, t = 1, 2, \dots, T\}$ , and  $y_{T+h}$  is the true value. RMSSE focuses on the expectation, which is consistent with the candidate methods in the forecasting pool.

The feature-based framework for intermittent demand is an extension of Fig. 10.1 by customizing a forecasting pool, a loss function based on RMSSE and time series features. The procedures of the aforementioned framework are formulated into the **fide** R package available at <https://github.com/lily940703/fide>.

### *Uncertainty Estimation*

The literature on the uncertainty estimation of feature-based time series forecasts is limited compared to point forecasting. The M4 forecasting competition (Makridakis et al., 2020) encouraged participants to provide point forecasts and prediction intervals (PIs). Among the submissions, Montero-Manso et al. (2020) computed the point forecasts using FFOMA and obtained the PIs by using a simple equally weighted combination of the 95% bounds of naïve, theta, and seasonal naïve methods. This approach ranked second in the M4 competition but

did not consider any time series features when calculating the interval forecasts. Wang et al. (2022b) applied meta-learners to explore how time series features affect the uncertainty estimation of forecasts, which is measured by PIs. GAMs (Hastie, 2017) are applied to depict the relationship between time series features and interval forecasting accuracies, making interval forecasts interpretable for time series features.

Wang et al. (2022b) proposed a feature-based framework for time series interval forecasting. The reference dataset used for training the algorithm is generated by GRATIS approach (Kang et al., 2020). The authors used a set of 42 features which are the same as the features in Montero-Manso et al. (2020). The mean scaled interval score (MSIS) is used to measure the accuracy of PIs. The calculation of MSIS can be stated as follows:

$$\text{MSIS} = \frac{1}{h} \frac{\sum_{t=n+1}^{n+h} (U_t - L_t) + \frac{2}{\alpha} (L_t - Y_t) \mathbb{1}\{Y_t < L_t\} + \frac{2}{\alpha} (Y_t - U_t) \mathbb{1}\{Y_t > U_t\}}{\frac{1}{n-m} \sum_{t=m+1}^n |Y_t - Y_{t-m}|}, \quad (10.9)$$

where  $Y_t$  are the true values of the future data,  $[L_t, U_t]$  are the generated PIs,  $h$  is the forecasting horizon,  $n$  is the length of the historical data, and  $m$  is the time interval symbolizing the length of the time series periodicity (e.g.,  $m$  takes the values of 1, 4, and 12 for yearly, quarterly, and monthly data, respectively).  $\mathbb{1}$  is the indicator function, which returns 1 when the condition is true and otherwise returns 0.

Based on the error measure of MSIS, Wang et al. (2022b) established the relationship between the mean values of  $\log(\text{MSIS})$  and the extracted features by using GAMs. The authors also selected a subset of appropriate methods for each time series tailored to their features from the method pool using an optimal threshold ratio searching algorithm. An R package **fuma** implements the aforementioned framework, which is available at <https://github.com/xqnwang/fuma>.

## CONCLUSION

In the era of big data, the information from the whole dataset is often used to improve forecasts. With the development of AI, a series of feature-based methods have sprung up. Scholars use meta-learning to describe the relationship between features and forecast model selection/combination. There is a general framework for feature-based methods, which contains

the model training and forecasting phases. Firstly, one needs to determine a set of diverse reference data to train the meta-learner. Generating synthetic time series with **gratis** provides an efficient way to generate time series with diverse and controllable characteristics. Secondly, we select a set of features used to describe the time series characteristics for forecasting. In addition to selecting appropriate features manually based on expert knowledge, the automation of feature extraction has gotten more attention in recent years. Advanced methods include time series imaging, calculation of forecast diversity and automatic feature selection procedures. Thirdly, deciding on a forecast pool is necessary, especially when there is a plethora of alternative methods. Accuracy, robustness, and diversity of the individual forecasts are the most critical factors that affect the pool's forecasting performance. A recently proposed algorithm named RAD can address the three factors simultaneously and identify an optimal subset from the initial forecast pool.

Based on the above reference set, selected features and trimmed forecast pool, a meta-learner can be trained based on features and forecast errors and obtain a forecast selection/combination model. The forecasting phase calculates the features for the new time series and gets the optimal method or combination weights through the pre-trained model. The feature-based framework is feasible with adjustable forecast pools, features, error measures, and meta-learners, which has been extended to practical issues, such as intermittent demand forecasting and uncertainty estimation.

However, one may also notice that the methods in this chapter are mostly for point forecasting with deterministic features. Very limited attention has been given to probabilistic time series forecasting based on features in the literature. More study is required for interpreting features and the associated weights. Solely using time series features to forecast is also a promising direction when data privacy is a concern in the big data era.

**Acknowledgments** Feng Li's research was supported by the National Social Science Fund of China (22BTJ028).

## REFERENCES

- Atiya, A. F. (2020). Why does forecast combination work so well? *International Journal of Forecasting*, 36(1), 197–200.
- Bagnall, A., Bostrom, A., Large, J., & Lines, J. (2017). *Simulated data experiments for time series classification part 1: Accuracy comparison with default settings*. arXiv preprint [arXiv:1703.09480](https://arxiv.org/abs/1703.09480)
- Bédubourg, G., & Le Strat, Y. (2017). Evaluation and comparison of statistical methods for early temporal detection of outbreaks: A simulation-based study. *PloS One*, 12(7), e0181227.
- Budescu, D. V., & Chen, E. (2015). Identifying expertise to extract the wisdom of crowds. *Management Science*, 61(2), 267–280.
- Cang, S., & Yu, H. (2014). A combination selection algorithm on forecasting. *European Journal of Operational Research*, 234(1), 127–139.
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.
- Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh-a python package). *Neurocomputing*, 307, 72–77.
- Collopy, F., & Armstrong, J. S. (1992). Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. *Management Science*, 38(10), 1394–1414.
- Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: The empirical structure of time series and their methods. *Journal of the Royal Society Interface*, 10(83), 20130048.
- Goldstein, D. G., McAfee, R. P., & Suri, S. (2014). The wisdom of smaller, smarter crowds. In *Proceedings of the fifteenth ACM conference on Economics and computation* (pp. 471–488).
- Hastie, T. J. (2017). Generalized additive models. In *Statistical Models in S* (pp 249–307). Routledge.
- Hyndman, R., Kang, Y., Montero-Manso, P., Talagala, T., Wang, E., Yang, Y., O'Hara-Wild, M., & Taieb, S. B. (2020). *tsfeatures: Time Series feature extraction*. <https://pkg.robjhyndman.com/tsfeatures/index.html>. R package version 1.0.2.
- Kang, Y., Cao, W., Petropoulos, F., & Li, F. (2022). Forecast with forecasts: Diversity matters. *European Journal of Operational Research*, 301(1), 180–190.
- Kang, Y., Hyndman, R. J., & Li, F. (2020). Gratis: GeneRAting TIme Series with diverse and controllable characteristics. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 13(4), 354–376.

- Kang, Y., Hyndman, R. J., & Smith-Miles, K. (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, 33(2), 345–358.
- Kegel, L., Hahmann, M., & Lehner, W. (2017). Generating what-if scenarios for time series data. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management* (pp. 1–12).
- Kolassa, S. (2016). Evaluating predictive count data distributions in retail sales forecasting. *International Journal of Forecasting*, 32(3), 788–803.
- Kourentzes, N., Barrow, D., & Petropoulos, F. (2019). Another look at forecast selection and combination: Evidence from forecast pooling. *International Journal of Production Economics*, 209, 226–235.
- Kück, M., Crone, S. F., & Freitag, M. (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In *2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 1499–1506). IEEE.
- Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10–12), 2006–2016.
- Lichtendahl, K. C., Jr., & Winkler, R. L. (2020). Why do some combinations perform better than others? *International Journal of Forecasting*, 36(1), 142–149.
- Li, F., Villani, M., & Kohn, R. (2010). Flexible modeling of conditional distributions using smooth mixtures of asymmetric student t densities. *Journal of Statistical Planning and Inference*, 140(12), 3638–3654.
- Li, L., Kang, Y., & Li, F. (2022a). Bayesian forecast combination using time-varying features. *International Journal of Forecasting*, 39(3).
- Li, L., Kang, Y., Petropoulos, F., & Li, F. (2022b). Feature-based intermittent demand forecast combinations: Accuracy and inventory implications. *International Journal of Production Research*, 1–16.
- Li, X., Kang, Y., & Li, F. (2020). Forecasting with time series imaging. *Expert Systems with Applications*, 160, 113680.
- Lotze, T. H., & Shmueli, G. (2009). How does improved forecasting benefit detection? an application to biosurveillance. *International Journal of Forecasting*, 25(3), 467–483.
- Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6), 1821–1852.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364.

- Mannes, A. E., Soll, J. B., & Larrick, R. P. (2014). The wisdom of select crowds. *Journal of Personality and Social Psychology, 107*(2), 276.
- Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting, 36*(1), 86–92.
- O'Hara-Wild, M., Hyndman, R., & Wang, E. (2022). Feasts: Feature extraction and statistics for time series. <http://feasts.tidyverts.org/>
- Petropoulos, F., Makridakis, S., Assimakopoulos, V., & Nikolopoulos, K. (2014). ‘Horses for courses’ in demand forecasting. *European Journal of Operational Research, 237*(1), 152–163.
- Prudêncio, R. B., & Ludermir, T. B. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing, 61*, 121–137.
- Rice, J. R. (1976). The algorithm selection problem. In *Advances in computers* (Vol. 15, pp. 65–118). Elsevier.
- Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning, 53*(1), 23–69.
- Shah, C. (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting, 13*(4), 489–500.
- Talagala, T. S., Hyndman, R. J., & Athanasopoulos, G. (2023). Meta-learning how to forecast time series. *Journal of Forecasting*. <https://doi.org/10.1002/for.2963>
- Talagala, T. S., Li, F., & Kang, Y. (2022). FFORMPP: Feature-based forecast model performance prediction. *International Journal of Forecasting, 38*(3), 920–943.
- Theodorou, E., Wang, S., Kang, Y., Spiliotis, E., Makridakis, S., & Assimakopoulos, V. (2022). Exploring the representativeness of the M5 competition data. *International Journal of Forecasting, 38*(4), 1500–1506.
- Thomson, M. E., Pollock, A. C., Önkal, D., & Gönül, M. S. (2019). Combining forecasts: Performance and coherence. *International Journal of Forecasting, 35*(2), 474–484.
- Vinod, H. D., & López-de Lacalle, J. (2009). Maximum entropy bootstrap for time series: The meboot r package. *Journal of Statistical Software, 29*, 1–19.
- Wang, X., Kang, Y., & Li, F. (2022a). Another look at forecast trimming for combinations: robustness, accuracy and diversity. arXiv preprint [arXiv:2208.00139](https://arxiv.org/abs/2208.00139)
- Wang, X., Kang, Y., Petropoulos, F., & Li, F. (2022b). The uncertainty estimation of feature-based forecast combinations. *Journal of the Operational Research Society, 73*(5), 979–993.
- Wang, X., Smith-Miles, K., & Hyndman, R. (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing, 72*(10–12), 2581–2594.

- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.  
<https://doi.org/10.1109/4235.585893>
- Wong, C. S., & Li, W. K. (2000). On a mixture autoregressive model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1), 95–115.
- Zhang, G. P., Patuwo, B. E., & Hu, M. Y. (2001). A simulation study of artificial neural networks for nonlinear time-series forecasting. *Computers & Operations Research*, 28(4), 381–396.

PART V

---

## Special Applications



## CHAPTER 11

---

# Deep Learning Based Forecasting: A Case Study from the Online Fashion Industry

*Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma,  
and Tim Januschowski*

## INTRODUCTION

Forecasting problems in the fashion industry and in particular, in its online variant, come with a particular set of challenges and downstream use-cases which we illustrate in our article. The arguably most peculiar challenge in the fashion industry stems from a fixed inventory assumption where a meaningful part of the inventory arrives at the beginning of

---

M. Kunz (✉) · S. Birr · M. Raslan · L. Ma · T. Januschowski

Zalando SE, Berlin, Germany

e-mail: [manuel.kunz@zalando.de](mailto:manuel.kunz@zalando.de)

S. Birr

e-mail: [stefan.birr@zalando.de](mailto:stefan.birr@zalando.de)

a season (having been ordered much ahead of the season) and it cannot be re-ordered throughout the season. Hence, for this non-re-orderable part of the assortment, forecasting is not primarily used for replenishment use-cases as is the case in say, grocery (e.g., Fildes et al., 2022). Instead, a primary use-case for forecasting is pricing/discounting (e.g., Li et al., 2020). Forecasting provides the input for a downstream mixed integer optimization problem that sets the prices with a notion of optimality in a classical forecast-then-optimize setting. In the online fashion industry, solving the pricing problem is a means to handle inventory risk management because pricing presents a major lever to reduce the risk to be overstocked at the end of the season. Complications that arise from the fixed inventory assumptions include the prominence of cold start problems (a large part of the catalogue is renewed each season) and short history. Other complications are commonalities also present in other online industries, namely a large catalogue with millions of stock keeping units (SKUs) which on the one hand results in a large overall scale of the forecasting problem and on the other hand in sparsity on the SKU level. We present examples and illustrations of the complications in a detailed (necessarily anecdotal) description of the data and the implied challenges for forecasting.

The main contribution of our article is the description of the approach that Zalando SE, a market-leading fashion retailer in Europe, has taken to address the forecasting problems that arise in pricing. Demand forecast subject to differing pricing levels poses the primary forecasting challenge that we aim to address. Given the complications described above, a natural choice is a flexible and hence, highly parameterized, global (Januschowski et al., 2020) forecasting model that allows to learn complex patterns across products and seasons in a data-driven way. We describe a formerly unpublished deep learning based forecasting model relying on the transformer (Vaswani et al., 2017) architecture. It has been in use at Zalando in

---

M. Raslan  
e-mail: [mones.raslan@zaland.de](mailto:mones.raslan@zaland.de)

L. Ma  
e-mail: [lei.ma@zalando.de](mailto:lei.ma@zalando.de)

T. Januschowski  
e-mail: [tim.januschowski@zalando.de](mailto:tim.januschowski@zalando.de)

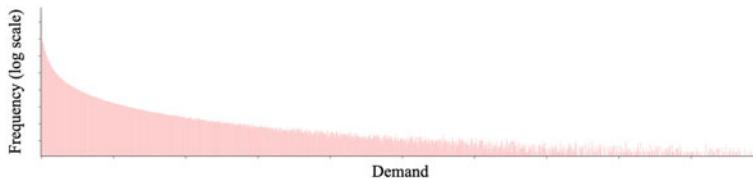
different versions since 2019 and hence is, to the best of our knowledge, among the first examples of attention-based forecasting models in production in industry.

Our article is structured as follows. We start by describing the data and associated covariates in section “[Data for Forecasting at Zalando: An Overview](#)”. In section “[Demand Forecast Model](#)”, we describe our model in detail and explain how we address the problems that arise in our context. We place a particular focus on the importance of the rich covariate structure available in our forecasting problems. The importance of price as a covariate is fundamental and we describe how we incorporate inductive biases to learn appropriate price elasticities into our model. In section “[Empirical Results](#)”, we include empirical results and compare our model performance against other approaches. We also try to explain why a transformer-based architecture performs better for our use-case, by testing for scaling laws. We discuss related work in section “[Related Work](#)”, where we mention advantages and limitations of our approach, contrasting our approach with publicly available methods. In section “[Practical Considerations](#)”, we comment on practical challenges such as monitoring forecasting accuracy in production, its down-stream usage and deciding how to re-train. We conclude in section “[Conclusion](#)”, where we also discuss avenues for future work that hopefully inspire future research.

## DATA FOR FORECASTING AT ZALANDO: AN OVERVIEW

Zalando is active in 25 European markets, its catalog comprises of >6500 brands with >50M active customers. This amounts to 1.6 million articles on the Zalando platform (as of Q3 2021). We are interested primarily in modelling the future demand of an article, where the demand is the quantity of items customers would buy in one time unit. Demand at Zalando follows a typical long-tail distribution, see Fig. 11.1. Note that we cropped the histogram on the right and cannot provide units for the x-axis and y-axis for confidentiality reasons. Most articles do not sell on most days, but there are a few high-selling products. Similar demand characteristics are available in other real-world examples (Laptev et al., 2017; Lim et al., 2021; Salinas et al., 2020).

Complementing the aggregate statistics, Fig. 11.2 gives examples for what individual time series look like. In Fig. 11.3, we show additionally the stock levels in red dashed lines for an example of an article that has a



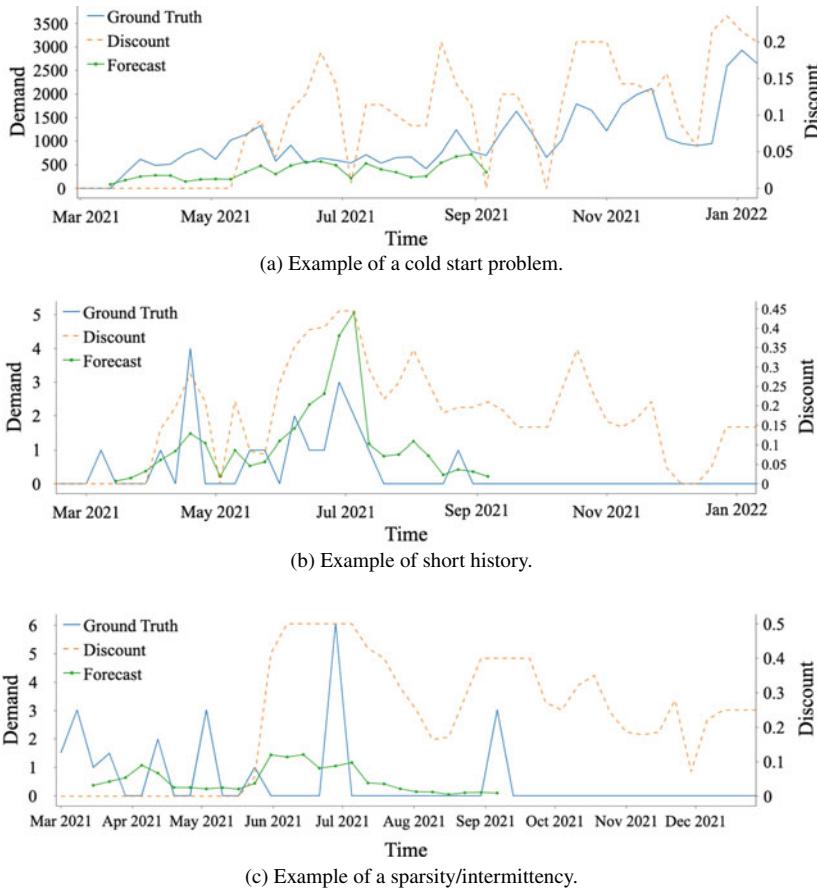
**Fig. 11.1** Histogram of demand (in log scale) per article: The histogram shows the heavy-tailed demand distribution. On the x-axis is demand starting from 0, on the y-axis the frequency of the demand occurrence over one time unit

complicated stock availability. These examples illustrate the difficulties in extracting patterns from the observational time horizon of a single article.

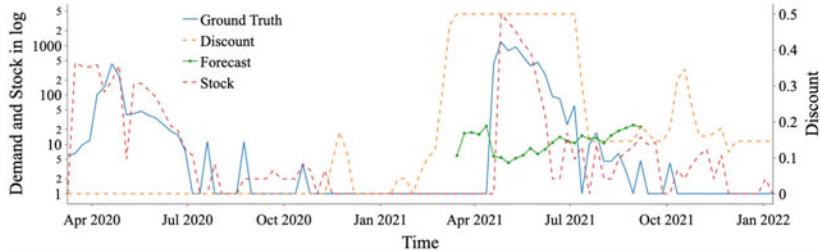
### Sales to Demand Translation

Note that, similar to other demand forecasting problems (e.g., Böse et al., 2017; Faloutsos et al., 2019; Fildes et al., 2022; Wen et al., 2017), we do not observe demand, but only sales. Demand corresponds to what the customers want to purchase, and materializes only when enough stock is available. In our case, we have a stock signal available which indicates when sales are strictly less than demand because no stock is available. While it is possible to handle such cases by marking the demand data as *missing* and let the forecast model handle this case (see Bohlke-Schneider et al., 2022 for an example), here we adopt a different approach by preprocessing our sales data and impute demand.

We can take advantage on the fact that in fashion, articles typically come in different sizes. Note that we do not desire to price differently different sizes, and so the forecasting task requires to predict demand of a single fashion item aggregated over all sizes. If all sizes of an article are available or *in stock* over one time unit, the demand of that article equals the materialized/observed sales. If some sizes are out of stock, the demand is in fact (partially) unobserved and we cannot directly obtain it from the sales data. To remediate this, we first define the demand  $q$  of an article  $i$  with  $k$  sizes as a random variable  $\mathbf{q}_i = \{q_{i1}, q_{i2}, \dots, q_{ik}\}$ . In order to infer the expected demand over all sizes  $n_i = \sum_{n=1}^k q_{in}$ , we assume  $\mathbf{q}_i$  to be multinomially distributed with  $\mathbf{q}_i \sim \text{Multinomial}(n_i, \mathbf{p}_i)$  and  $\mathbf{p}_i = \{p_1, p_2, \dots, p_k\}$  the discrete probability distribution over the  $k$  sizes of article  $i$ . If we further assume to know  $\mathbf{p}_i$ , we can compute  $n_i$  based on



**Fig. 11.2** Examples for complications occurring in practice: The dashed orange line gives discount level (the higher, the more discount), blue is observed (ground truth) demand and green with markers is the forecast. Note that everything left to the first green dot is the available demand history. We have hence articles with little to no history, see Fig. 11.2a and b as well as sparse time series, Fig. 11.2c



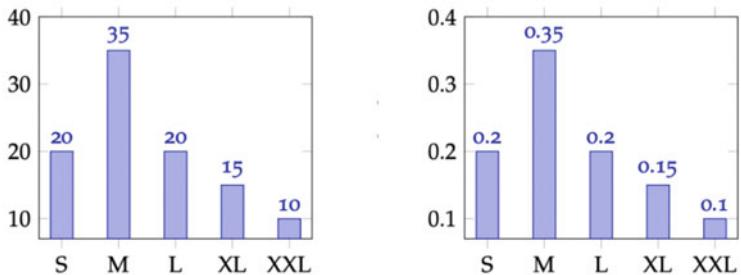
**Fig. 11.3** Example for a challenging stock situation: The red dashed line represents stock levels over time. The article was out of stock for a longer time period until it was replenished in April 2021. It can be seen that the ground truth is highly correlated with stock availability of an article which provides additional modelling challenges as discussed in section “[Sales to Demand Translation](#)”

a partial observation  $\{q_{i1}, q_{i2}, \dots, X_{ij}, \dots, q_{ik}\}$ , with  $X_{ij}$  representing the missing demand observations. This procedure only requires to learn  $\mathbf{p}_i$  from observational data in intervals with no stock-outs, assuming  $\mathbf{p}_i$  to be time invariant. We then use the learned  $\mathbf{p}_i$  to translate sales to demand, and we use this imputed demand in model training. Note that, when adopting such a preprocessing step, special care has to be taken by not evaluating our model on extrapolated data, but rather exclude this data from evaluation. In Fig. 11.4, we depict the process described above and Fig. 11.5 visualizes the result of our model applied to a specific article over time.

### Description of Covariates

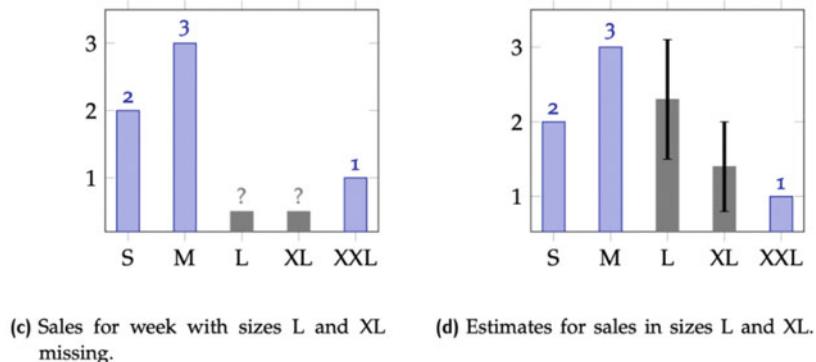
The covariates that we use in our forecasting model can be grouped into the following categories: *static-global*: time-independent and market-independent; *dynamic-global*: time-dependent and market-independent; *dynamic-international*: time-dependent and market-specific; and *static-international*: time-independent and market-specific. We give a non-exhaustive list for the covariates in use, described in Table 11.1 in more detail for illustration purposes.

The *static-global* covariates Brand and Commodity Group categorize an article. The later provides information about the part of the assortment (e.g., Men—Sportswear). They do not change over time and across countries. The Black Price is the recommended retail price and the initial



(a) Total observed sales over all weeks with full size availability.

(b) Probability distribution over sizes ( $\bar{p}$  parameter of Multinomial).

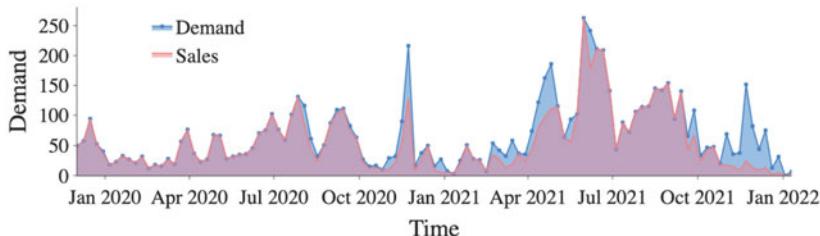


(c) Sales for week with sizes L and XL missing.

(d) Estimates for sales in sizes L and XL.

**Fig. 11.4** Sales to demand translation for an article with 5 sizes: In a, historical sales observations of an article over weeks with all sizes available are aggregated. Based on a, we obtain in b, the article's empirical probability distribution over sizes. In c, the weekly demand of an article with missing sizes L and XL is illustrated. The unobserved demand for L and XL is inferred in d, given the observed demand of the other sizes for that week and the empirical distribution computed in b

price an article receives. It is the reference price for every discount applied. We use its logarithm to prevent scaling issues typical in global forecasting models (Januschowski et al., 2020). Black Price and Discount are *international* covariates, containing potentially different values in different



**Fig. 11.5** Demand vs sales for a single article. In weeks with full availability, sales and demand overlaps. If a size or the full article becomes unavailable, demand becomes an estimate larger than the observed sales

**Table 11.1** Examples for covariates available for the forecasting problem at Zalando

Feature	Dim	Category	Value Type	Transformation	available in future?
Brand	10	Static-global	Categorical	Embedded	y
Commodity Group	10	Static-global	Categorical	Embedded	y
Discount	14	Dynamic-int'l	Numeric	-	n
Black Price	14	Static-int'l	Numeric	Logarithm	y
Sales	14	Dynamic-int'l	Numeric	Logarithm	n
Stock	1	Dynamic-global	Numeric	Logarithm	n
Stock Uplift	1	Dynamic-global	Numeric	Logarithm	n

countries. The Black Price of an article does not change over time, but Discount might change which makes it *dynamic*.

Note that not all covariates are available for the forecast horizon. For example, Stock or Discount are only available historically. We include them because they either help to “explain away” certain historical effects or because we set them explicitly in the forecast horizon to obtain what-if forecasts. Other covariates, like the Stock Uplift, are only available as an estimate for the future time horizon. The Stock Uplift expresses for each time unit the increase in stock of an article, caused by supplier deliveries or customer returning their purchase. This value is available for the past but only partially available for the future and needs to be estimated based

on supplier delivery data. Both Stock and Stock Uplift are *global*, because stock is shared across countries.

## DEMAND FORECAST MODEL

The demand forecasting model used within Zalando Pricing is a global (Januschowski et al., 2020; Montero-Manso & Hyndman, 2021) forecasting model. This means that a single forecasting model is trained over the entire Zalando article<sup>1</sup> assortment and used to provide article specific predictions on a weekly time resolution for a horizon of 26 weeks. Additionally, our forecast is used to decide on the discounts applied to each article: we need to forecast future demand for multiple scenarios. Before we describe the forecasting approach in detail, we first formalize the forecasting problem.

### Problem Formalization

For any time series  $x$ ,  $x_{0:T}$  is short-hand for  $[x_0, x_1, \dots, x_T]$ . The observational time series data of an article  $i$  at time  $t$  starting at 0 is given by  $\{q_{i0:t}, d_{i0:t}, z_{i0:t}\}$ , where  $q$  denotes the demand,  $d$  corresponds to the discount, which is the percentage of price reduction relative to the article's recommended retailer price; and  $z$  is a set of article-specific covariates. Our object of interest is

$$P(q_{i,t+1:t+h} | q_{i,0:t}, d_{i,0:t+h}, z_{i,0:t+h}; \theta), \quad (11.1)$$

that is the probability distribution of demand in the forecast horizon  $t + 1 : t + h$  conditioned on (among others) discounts in the forecast horizon. We are interested in identifying appropriate  $\theta$ . Note that (11.1) is a simplification. More generally, we should be interested on the one hand in a multi-variate version so that we model cross-article effects and, on the other hand, we should actually be interested in

$$P(q_{i,t+1:t+h} | \text{do}(d_{t+1:t+h}), q_{0:t}, d_{0:t}, z_{0:t+h}; \theta), \quad (11.2)$$

where the do operator denotes the intervention of setting a price/discount to a certain level (see e.g., Pearl, 2009 for the notation and

<sup>1</sup> A sellable article offered in the Zalando fashion store.

additional background). This is because we are interested in taking downstream decisions on prices so the (causal) relationship between price/discount and demand is of fundamental importance. We leave a more comprehensive causal treatment for future work and mention that in practice, we approximate (11.1) by a point forecast and point to (Gouttes et al., 2021) for a discussion of a probabilistic extension of the model described here. However, we remark that (11.1) is a simplification of our actual model in the sense that we provide a multi-variate forecast because we forecast all markets in which Zalando is active in simultaneously. So our demand time series actually has an additional index per market. However, for ease of exposition, we do not further elaborate on this complication.

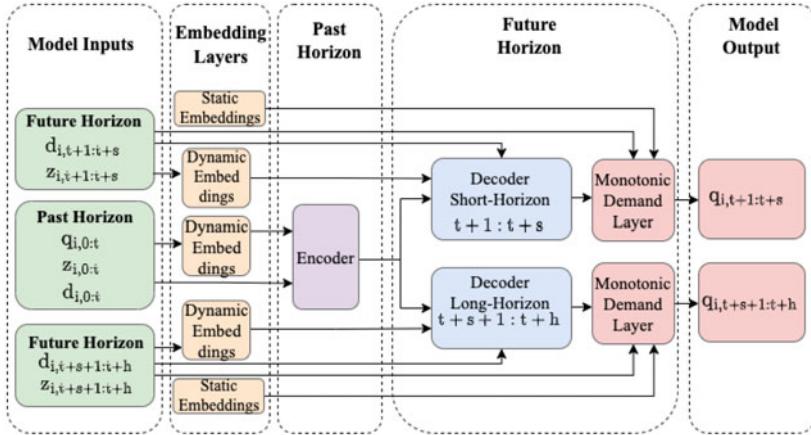
### *Model Architecture*

For the demand forecast model, we rely on an encode/decoder (Sutskever et al., 2014) approach. This is primarily motivated by the fact that we have some covariates that are not available in the forecast horizon and others that are available only for the future, see Table 11.1. The encoder/decoder architecture is particularly amendable for this setting (see e.g., Faloutsos et al., 2019; Wen et al., 2017).

For the model architecture itself, we base it on the standard Transformer architecture and we depict it in Fig. 11.6. We have additional embedding (Mikolov et al., 2013) layers for some of the covariates. Our main modelling innovations are twofold. First, we specialize our decoder into a short- and a long-term horizon. This is because our downstream application requires particular focus on the first forecast weeks for which the pricing decisions are taken, whereas the further out future only plays a secondary importance. We further adapt the training scheme to account for the focus on the first weeks. We discuss this in section “Training”. Second, we need to control the relationship between price and demand closely given the primary usage of the forecasts in pricing. We achieve this by enforcing a monotonic relationship between price and demand which we parameterize flexibly as a piece-wise linear function (section “Monotonic Demand Layer”).

### *Input Preparation*

In the forward pass, the model receives the data of a single article shown as Model Inputs in Fig. 11.6. All features are stacked into a single input



**Fig. 11.6** Demand Forecaster Architecture: Encoder handles the observational time series, decoder incorporates the future covariates. Future discounts are directly provided to the output (Monotonic Demand) layer, skipping decoder and encoder

tensor of dimension  $\mathbb{R}^{t \times u \times v}$ , where  $t$  is the time dimension,  $u$  is the batch size, and  $v$  is the dimension of the covariate vector.

The high dimensional but sparse categorical covariates are passed through the embedding layers before being forwarded to the encoder, decoder and monotonic demand layer. We distinguish between two types of embeddings, dynamic and static. Dynamic embeddings are time-dependent and provided as input to the encoder and decoder. Static embeddings are time-independent and provided as additional input to the monotonic demand layer. The temporal structure of the time series is represented by positional encoding, introduced in section “[Positional Encoding](#)”.

The decoder receives all covariates associated with the forecasting time horizon, the associated embeddings and the output of the encoder. Encoder and decoder consist of various multihead attention layers (Bahdanau et al., 2014; Vaswani et al., 2017), to represent temporal correlations across the covariates. Instead of feeding future discounts into the decoder, we directly provide them to the monotonic demand layer, see section “[Monotonic Demand Layer](#)”, which allows us to predict demand as a monotonically increasing function of discount.

### Encoder

The task of the encoder is to learn a representation of the observational article data, optimized for predicting future demand. The encoder model architecture is based on multi-head attention (Vaswani et al., 2017). In the following, we recall the concepts and explain how to translate them to the time series setting. In a first step, we project the input data  $\{d_{0:t}, q_{0:t}, z_{0:t}\}$  into three  $(Q, K, V)^2$  tensors. Therefore, for each article  $i$ , the data is transformed into the matrix  $\eta_{i,0:t}$ :

$$\eta_{i,0:t} = \begin{bmatrix} q_{00} & q_{01} & \cdots & q_{0t} \\ d_{10} & d_{11} & \cdots & d_{1t} \\ z_{20} & z_{21} & \cdots & z_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ z_{v0} & z_{v1} & \cdots & z_{vt} \end{bmatrix}, Q = \phi_q(\eta_{i,0:t}), K = \phi_k(\eta_{i,0:t}), V = \phi_v(\eta_{i,0:t}) \quad (11.3)$$

Each column in  $\eta_{i,0:t}$  represents a covariate vector with demand, discount, and additional covariates of an article  $i$  at time  $T \in [0, \dots, t]$ . The matrix  $\eta_{i,0:t}$  is then multiplied by three learned weight matrices, implemented as linear layers  $\phi_q, \phi_k, \phi_v$ . Scaled dot product attention (11.4) and multi-head attention (11.5) is then applied to  $Q, K$ , and  $V$ , following (Vaswani et al., 2017).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (11.4)$$

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \\ \text{head}_j &= \text{Attention}(QW_j^Q, KW_j^K, VW_j^V) \end{aligned} \quad (11.5)$$

In the original transformer architecture,  $d_{model}$  represents the embedding dimension of an embedded word vector. Applied to the time series data set in 11.3,  $d_{model}$  is defined by the number of rows in  $\eta_{i,0:t}$ , or expressed differently, by the covariate vector of dimension  $v$ , so  $v = d_{model}$ . Similar as in the original transformer architecture, the number of

<sup>2</sup>  $Q$  = Query,  $K$  = Key,  $V$  = Value as in the standard transformer nomenclature; note that this clashes with the notation that we introduced before unfortunately, but it is standard in the transformer literature.

attention heads  $h$ , the dimension of the linear layers  $d_k$  and the dimension of the embedding layers were chosen to match  $d_{model} = h \cdot d_k$ . The linear layers  $\phi_q, \phi_k, \phi_v$  were specified with same input and output dimension to produce  $Q, K$  and  $V$  tensors of shape  $h \times (t + 1) \times d_k$ . The remaining operations follow (11.4 and 11.5). Note that the computation of the  $h$  attention heads allows parallelization and is most efficiently implemented by applying vectorized tensor operations (Paszke et al., 2019). The encoder contains multiple stacked multi-head attention layers with two sublayers and residual connections (He et al., 2016) as specified in (Vaswani et al., 2017). Dropout (Srivastava et al., 2014) and Layer Normalization (Lei Ba et al., 2016) is applied on the output of each sublayer.

### *Positional Encoding*

Positional encoding is a fundamental part of the original transformer architecture. For each position in the input embeddings, a positional encoding of dimension  $d_{model}$  is computed and summed up with the corresponding word embedding at this position (Vaswani et al., 2017). This approach is not transferable to the time series data introduced in section “[Data for forecasting at Zalando: An Overview](#)” because of the heterogeneous covariate structure. Instead, a positional encoding feature with embedding dimension  $e_{dim}$  is computed and integrated as an independent covariate to  $\eta_{i,0:t}$ .

$$\begin{aligned} pos_{0:t} &= [pos_0^T \ pos_1^T \ \cdots \ pos_n^T \ \cdots \ pos_t^T] \\ pos_n &= [\sin(p_{n,0}), \cos(p_{n,1}), \dots, \sin(p_{n,e_{dim}-2}), \cos(p_{n,e_{dim}-1})] \\ p_{n,m} &= 2 \cdot \pi \cdot f_m \cdot n \\ f_m &= \frac{2 \cdot m + 1}{e_{dim} \cdot t_{dim}} \end{aligned} \tag{11.6}$$

The positional encoding feature is defined by  $pos_{0:t}$  in (11.6). Each position in  $pos_{0:t}$  is a sequence of sine and cosine values computed at different frequencies  $f_m$  for each embedding dimension in the positional encoding. Each frequency is defined relative to  $t_{dim} = 52$ , representing annual cycles given weekly time resolution.

### *Padding and Masking*

The original transformer architecture applies zero padding and masking to sequences with variable length in order to reach the defined input embedding dimension and to avoid a look-ahead in the decoder. In time series forecasting, only the time dynamic covariates need to be zero padded and masked which becomes especially important if slicing is applied in order generate multiple training samples out of one observational time series. Masking is then applied to the result of the dot product of the  $Q$  and  $V$  tensors in order to prevent the padded dimensions influencing the attention scores. In the case of the demand forecasting model, not only zero padded values are masked out but also all data points with zero stock availability. This is motivated by issues inferring demand on article data with zero stock availability.

### *Decoder*

The task of the decoder is to compute for each time unit in the forecast horizon a future state. The matrix  $\eta_{i,t+1:t+h}$  11.7 represents the input to the decoder. Each column vector  $\eta_{i,T}$  in  $\eta_{i,t+1:t+h}$  contains the known future covariates of an article  $i$  in future week  $T \in [t+1, \dots, t+h]$  and the last observed discount  $d_t$  and demand  $q_t$  at time  $t$ . In order to compute the future state at time  $T$ , the decoder receives the encoder's output  $\gamma_{i,0:t}$  and the future covariate vector  $\eta_{i,T}$  (11.8) as input. Similar to the encoder, the decoder contain multi-head attention layers but with a specific attention mechanism (11.9) applied.

$$\gamma_{i,0:t} = \begin{bmatrix} e_{00} & e_{01} & \cdots & e_{0t} \\ e_{10} & e_{11} & \cdots & e_{1t} \\ e_{20} & e_{21} & \cdots & e_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ e_{k0} & e_{k1} & \cdots & e_{kt} \end{bmatrix}, \quad \eta_{i,t+1:t+h} = \begin{bmatrix} d_{0t} & d_{0t} & \cdots & d_{0t} \\ q_{1t} & q_{1t} & \cdots & q_{1t} \\ z_{2,t+1} & z_{2,t+2} & \cdots & z_{2,t+h} \\ \vdots & \vdots & \ddots & \vdots \\ z_{k,t+1} & z_{k,t+2} & \cdots & z_{k,t+h} \end{bmatrix} \quad (11.7)$$

$$Q = \phi_q(\eta_{i,T}), \quad K = \phi_k(\gamma_{i,0:t}), \quad V = \phi_v(\gamma_{i,0:t}) \quad (11.8)$$

In (11.8), the decoder input is passed through a linear layer  $\phi$ , similar as in (11.3). Note that the dimension of  $\eta_{i,T}$  and  $\gamma_{i,0:t}$  are different, causing  $Q$  to be different in dimension compared to  $K$  and  $V$ . This is solved by repeating values of  $Q$  to align in dimension, before stacked together and passed through linear layer  $\tau$  computing the inner product across past and

future.

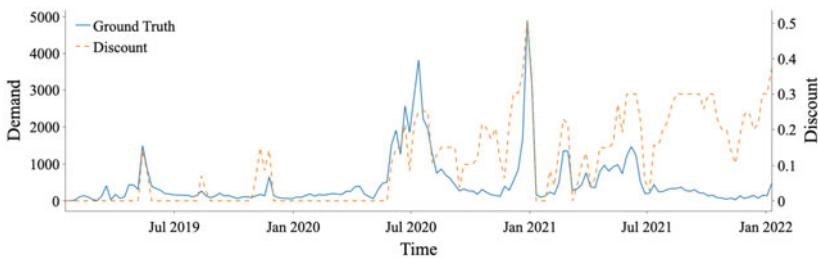
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\tau(Q, K)}{\sqrt{d_k}}\right)V \quad (11.9)$$

The slightly changed attention formula shown in (11.9) is applied for each column vector in  $\eta_{i,t+1:t+h}$ , computing the future states in the forecast horizon  $t + 1 : t + h$ . Different to the original transformer architecture, the decoder is non-autoregressive, allowing to compute each future state in parallel. This architectural choice was necessary to enforce independence<sup>3</sup> between forecast weeks. Similar as in the encoder, normalization, dropout, and residual connections are applied. The decoder output  $\kappa_{i,t+1:t+h}$  is passed with the encoder state  $\gamma_{i,0:t}$  and the future discounts  $d_{i,t+1:t+h}$  to the Monotonic Demand Layer in order to predict the future demand  $q_{i,t+1:t+h}$ .

#### *Monotonic Demand Layer*

The demand forecast model represents future demand as a function of future discount  $d_{i,t+1:t+h}$ . This requires not only to learn the temporally dependent dynamics of the demand time series but also the demand response function w.r.t. discount.

Figure 11.7 shows demand and discount time series, illustrating the positive correlation between the two quantities. Although it might be obvious that the demand always increases with an increase in discount,



**Fig. 11.7** Demand and discount time series of an article, illustrating the positive correlation between discount (orange dashed) and demand (blue)

<sup>3</sup> The independence assumption between future weeks was made in order to simplify the downstream pricing models, consuming the output of the demand forecaster.

the underlying causal mechanism is subject to confounding, resulting in counter-intuitive demand and discount curves. In order to address this issue, we model demand  $q_{i,t+n}$  of future discount  $d_{i,t+n}$  as a piece-wise linear and monotonically increasing demand response function in the monotonic demand layer.

The domain of the discount-dependent function shown in Fig. 11.8 is divided into segments of equal width. Each segment contains a linear function, specifying the increase in demand relative to the black price demand. We assume the following parameterization of the demand response function  $\xi$ :

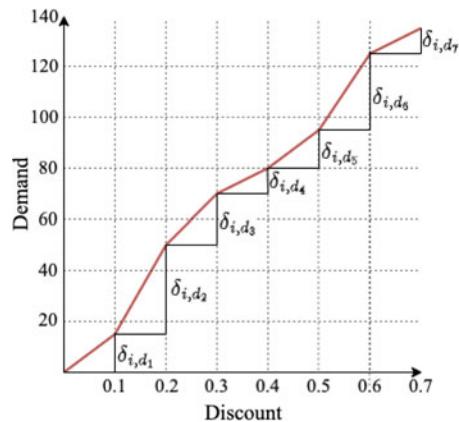
$$\begin{aligned}\xi(d_{i,t+n}) &= \hat{q}_{i,t+n} + \sigma_{i,t+n} \cdot \left( \sum_{n=1}^m \delta_{i,d_n} + d_{i,t+n} \cdot \frac{\delta_{i,d_{m+1}}}{0.1} \right) \\ d_{i,t+n} &\in [0, 0.7], m = \lfloor 10 \cdot d_{i,t+n} \rfloor\end{aligned}\quad (11.10)$$

The monotonic demand layer parameterizes the demand function defined in (11.10) via two feed forward neural networks,  $\phi_0$  and  $\phi_1$

$$\begin{bmatrix} \delta_{id_1} \\ \delta_{id_2} \\ \vdots \\ \delta_{id_7} \end{bmatrix} = \phi_0(\gamma_{i,0:t}), \begin{bmatrix} \hat{q}_{i,t+n} \\ \sigma_{i,t+n} \end{bmatrix} = \phi_1(\kappa_{i,t+n}). \quad (11.11)$$

The input to  $\phi_0$  and  $\phi_1$  is the output of encoder and decoder,  $\gamma_{i,0:t}$  and  $\kappa_{i,t+n}$ . The slopes  $\delta_{id_1}, \dots, \delta_{id_7}$  only depend on the encoder state,

**Fig. 11.8** Piecewise linear monotonic demand response function as defined in (11.10), x-Axis: the discount level, y-axis: The increase in demand under the applied discount. The  $\delta$  values represent the change in demand for a 10% point change in discount



resulting in the same values for all future weeks. Black Price (see Table 11.1) demand  $\hat{q}_{i,t+n}$  and scale parameter  $\sigma_{i,t+n}$  are dependent on the decoder output column  $\kappa_{i,t+n}$ , indexed with  $t+n$ , similar as the future discount  $d_{i,t+n}$ . The Softplus<sup>4</sup> function ensures that the output of  $\phi_0$  and  $\phi_1$  results in a monotonically increasing demand function w.r.t. discount  $d_{i,t+n}$  by construction. The monotonicity is a standard assumption in econometrics (see e.g., Phillips, 2021). The final demand prediction at the future discount level  $d_{i,t+n}$  is then the simple computation of (11.10), given the parameterization of  $\xi$  by  $\phi_0$  and  $\phi_1$ . In order to provide demand predictions  $q_{i,t+1:t+h}$  over the full forecast horizon  $t+1 : t+h$ , the steps are repeated for all future discounts  $d_{i,t+1:t+h}$  and the corresponding decoder output columns in  $\kappa_{i,t+1:t+h}$ .

### Near and Far Future Forecasts

In our approach, we decompose the forecast problem into near future and far future. The near future performance is of particular importance to our pricing use-case, hence we specialize our model architecture for this use-case by having one decoder and one monotonic demand layer for each of the two forecast horizons. We set 5 week as the threshold for near future, and 5–20 week forecast will be considered as far future.

During training, we first train for seven epochs the decoder and monotonic demand layer for near future and freeze the decoder and demand layer for the far future. Then we freeze the components serving for near future and only train one final epoch for far future. We chose this specific training procedure to reduce training time, since the backpropagation of the error over the long forecast horizon turned out to be a significant cost factor. The near future loss only computes for the first 5 weeks, i.e., the near future horizon; the far future loss is covering all the 20 weeks. Note that the model is used to predict for 26 weeks ahead using the far future decoder and demand layer, depending on extrapolating beyond the training horizon.

In general, for forecasting tasks the further we forecast into the future the more deterioration there will be in the forecast. From Fig. 11.9, we can see that the weighted  $\ell_2$  norm, which we use in practice as an evaluation metric, is much smaller on average in the near future part compared

<sup>4</sup> Smoothed out version of the ReLU (Nair & Hinton, 2010) function.

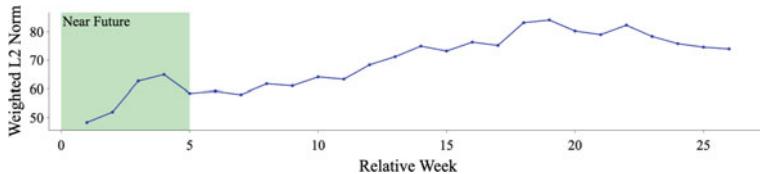


Fig. 11.9 Demand model accuracy over a 26 weeks forecast horizon

to the far future. The weighted  $\ell_2$  norm increases fast in the far future, which is expected. The far future is mainly used for long-term effect and to estimate trend, therefore the accuracy requirement for far future is not as high as the near future.

### *Training*

In our training procedure, we follow mostly standard deep learning approaches. For the loss function that we use to train our model, we note that in our downstream price optimization applicate, we are primarily interested in maximizing expected profit and profit is a (mostly) linear function. The linearity of the profit function allows us to work directly with the mean of (11.1). Hence, we use a loss function closely resembling the  $\ell_2$ -loss:

$$L = \sum_{j \in M} (v(\hat{q}_j) - v(q_j))^2 \quad (11.12)$$

where  $q$  is the demand and  $\hat{q}$  the predicted demand and  $v(x) = 1 + x + x^2/2 + x^3/6$  is the third-order Taylor approximation of  $e^x$ . This showed a stabilizing effect in the beginning of the training since the model became less sensitive to large errors. Note that we need to exponentiate, since we apply Box-Cox log transformation to our targets, see Table 11.1. Index  $j$  ranges over all markets  $M$ . The loss is not computed on the observed sales but on the inferred demand (see section “Sales to Demand Translation”). Recall from section “Near and Far Future Forecasts” that we place a special emphasis on the near future, since pricing decisions are updated frequently and only prices for the next time unit will materialize. We train the encoder and short-term decoder jointly for most of the epochs. In the last epochs of training, we freeze the encoder and only train the decoder for predicting the remaining far future weeks. This

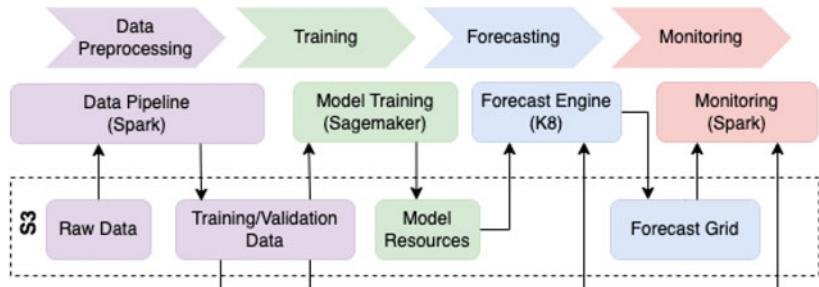
has the advantage to focus the encoder on the more important near-term forecasting problem but also has computational upsides as we do not need to train the high-dimensional decoder for the long term.

### *Prediction*

The demand forecast model generates article-level demand predictions for 14 different countries in the price range of 0 to 70% discount in 5% point discount steps. The output of this large-scale prediction run is a so-called *demand grid*  $\Upsilon \in R^{a \times c \times t \times d}$ . The demand grid  $\Upsilon$  spans over  $t = 26$  future weeks, up to  $a = 1 \times 10^6$  articles,  $c = 14$  countries, and  $d = 15$  discount levels. All predictions are generated via the global demand forecasting model introduced in section “[Near and Far Future Forecasts](#)”. In order to produce article-specific predictions, the model first loads the specific features of one article like demand and price history as well as categorical features and other covariates [11.1](#). The model then predicts future demand for this specific article given the future discount level. Put differently, we answer the question: what would be the demand  $q$  of an article  $i$  in future week  $t$  if we keep everything impacting demand constant but only change the price of the article by applying discount  $d$ . This makes it inherently a causal forecasting ([Bica et al., 2020](#)) problem as described in [\(11.2\)](#). The demand grid results in a large data batch with up to  $5.5 \times 10^9$  records and demand forecasts over the full range of future discount levels. Based on the discount-dependent demand forecast grid, the downstream price optimizer then selects an article-specific future price trajectory that maximizes the achievable profit of the individual article in the bounds of certain business rules like stock levels and maximum discount steps.

## EMPIRICAL RESULTS

The model is trained on the full Zalando article assortment, processing 4800 samples per batch. We choose 6 attention layers each in the encoder and decoder. Each multi-head attention layer contains  $h = 23$  attention heads. Including the heads, the dimension of the  $Q$ ,  $K$ , and  $V$  tensors is



**Fig. 11.10** Forecasting pipeline set-up in production

$h \times (j + 1) \times d_k$  with  $j = 51^5$  and  $d_k = 4^6$ . For the monotonic demand layer, we choose seven segments of equal size 0.1. For the hyperparameter  $\gamma$  in the loss (11.12), we choose the demand share of the associated market. In total, the model contains approximately  $5.3 \times 10^7$  trainable parameters.

A standard time to train the model is about 45 h on a four GPU instance (ml.g4dn.12xlarge) at a cost of approximately 275 US\$. We use a standard forecasting system set-up combining Spark and Amazon Sage-Maker (Böse et al., 2017; Liberty et al., 2020; Zaharia et al., 2016), depicted in Fig. 11.10. K8 refers to Kubernetes and S3 is the Amazon Web Service Storage Service.

### Accuracy Metrics

For measuring the accuracy of our demand forecast, we need metrics that fulfill the following criteria:

1. comparable between data sets with different number of articles
2. weighted by how valuable an article is
3. working well with our downstream optimization task.

<sup>5</sup>  $j$  is the observational time horizon and selected to cover a complete year starting at index 0, considering weekly time resolution.

<sup>6</sup>  $d_k$  is a hyperparameter defining the dimension of  $Q$ ,  $K$ ,  $V$ .

We define the *demand error*

$$D_{T,h} = \sqrt{\frac{\sum_i \sum_{T=t+1}^{t+h} b_i (\hat{q}_{i,T} - q_{i,T})^2}{\sum_i \sum_{T=t+1}^{t+h} b_i q_{i,T}^2}} \quad (11.13)$$

and the *demand bias*

$$B_{T,h} = \frac{\sum_i \sum_{T=t+1}^{t+h} b_i (\hat{q}_{i,T} - q_{i,T})}{\sum_i \sum_{T=t+1}^{t+h} b_i q_{i,T}} \quad (11.14)$$

as the custom metrics to assess the accuracy of our forecasting problem. Here,  $t$  is the last timepoint in the training set,  $h$  denotes the forecast horizon,  $\hat{q}_{i,T}$  is the prediction for article  $i$  at timepoint  $T$ ,  $q_{i,T}$  is the corresponding true demand and  $b_i$  is the black price of article  $i$ .

The demand error  $D_{T,h}$  measures how large our forecasting error is on a single article level. We weight this by black price  $b_i$  to place a higher importance on more expensive articles as a proxy for the overall value for Zalando. As these typically generate more profit per sale, an error in the demand forecast for these articles is more costly for us. Scaling ensures that even in our setting, where the set of forecasted articles and sold changes from week to week, we obtain comparable values. In contrast, the demand bias indicates whether we are on average over or underpredicting while using the same weighting and scaling as the demand error.

We also experimented with (weighted and scaled) versions of more common accuracy metrics, like MAPE and MAE, and ran simulations how this changes affect the downstream optimization task. It turns out that outliers are an important factor for the performance so that a metric that resembles the RMSE more closely will have the best result with respect to the downstream pricing application. This feature is also the biggest drawback of our demand error metric as we observe that the error is largely driven by a small portion of the data. Especially, forecast accuracy on articles with low and sparse demand (which is the majority of our articles) is not adequately addressed by these metrics.

### *Model Benchmarking*

To illustrate accuracy of our transformer model, we present a part of the empirical results we gathered. In particular, we compare the model described above with a naive forecast that simply repeats

the last observation, a tree-based model (Januschowski et al., 2022) using LightGBM (Ke et al., 2017) and the autoregressive recurrent neural network DeepAR (Salinas et al., 2020) as implemented in GluonTS (Alexandrov et al., 2019) for which we used the mean forecast as a point forecast. For the experiments, we chose 10 weeks in 2022, using all sales data from Germany from the category Textil. In Zalando, this is the largest category, and performance here is highly indicative of the performance on the whole assortment. We focus primarily on the performance of the first weeks forecast. Table 11.2 shows the average metrics in the given weeks, along our own metrics defined as in section “[Accuracy Metrics](#)”, we also calculated more widely used metrics, namely RMSE and MAPE, where, to deal with zeros, we added 10 to both predicted and true demand values.

Comparison of our transformer models with commonly used alternative approaches and a naive baseline. The metrics are calculated over 10 different weeks in 2022 and measure the performance of the predictions for the first week. We report here the mean and standard deviation over the 10 weeks.

We observe that our transformer model performs best in terms of demand error and RMSE. While tree based models seem to be a close competitor in terms of demand error and were much faster to train, we saw a significantly higher bias. In terms of demand bias and MAPE, both Deep Learning models are very close to each other, but for RMSE again lightgbm and our transformer model have a clear advantage. It is worth mentioning that the naive baseline performs quite competitively on the one week forecast. A more detailed analysis reveals that especially for high-selling articles using the naive forecast poses a strong baseline. Obviously during periods of high variation, as they happen at the beginning and the

**Table 11.2** Model Performance

<i>Forecaster</i>	<i>Demand Error</i>		<i>Demand Bias</i>		<i>RMSE</i>		<i>MAPE</i>	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Naive Forecaster	0.603	0.135	0.079	0.161	11.754	4.038	0.088	0.011
lightgbm	0.489	0.066	0.296	0.127	9.737	2.941	0.142	0.005
DeepAR	0.593	0.126	0.090	0.156	12.443	4.224	0.092	0.011
Transformer	<b>0.449</b>	0.04	<b>-0.047</b>	0.064	<b>9.588</b>	2.725	<b>0.081</b>	0.007

end of sales events, the naive forecast becomes almost useless. During the end of season sale in 2022, for example, we observed a demand error of 0.877 for the naive forecaster while our transformer model reached a much more stable demand error of 0.491.

### *On the Benefits of Transformer-based Forecasting: First Results*

In the overall literature on forecasting, conclusive evidence of outsized gains in predictive accuracy of transformer-based models is not yet available. On the one hand, results on publicly available data sets are positive (Lim et al., 2021), but not as overwhelmingly positive as in natural language processing (NLP) where transformers dominate. Careful comparisons with strong baselines (e.g., Borchert et al., 2022; Sun & Boning, 2022) place additional scrutiny on these results. We remark that transformer-based forecasting approaches tend to distinguish themselves by side-aspects such as long-term forecasting or interpretability whereas in NLP, the improvements in the core task are outsized.

On the other hand, on closed data sets, convincing evidence exists for the superiority of transformer-based forecasting models (e.g., Eisenach et al., 2020; Yang et al., 2022). Our results above point into a similar direction. So, a natural question to ask is whether (and how) the success of transformer-based models can be associated to the nature of these closed source data sets and in particular the abundance of data available for companies like Zalando. To make progress in answering this question, we considered an often-cited reason for the success of transformers, so-called scaling laws (e.g., Brown et al., 2020; Kaplan et al., 2020). These scaling laws state that the predictive performance on a test set continues to increase for transformers along various dimension including, for example, the size of the training set.

Concretely, we attempt to replicate scaling laws for transformers in forecasting (e.g., in Kaplan et al., 2020, Figure 1) and whether we can observe level effects (see Fig. 11.11).

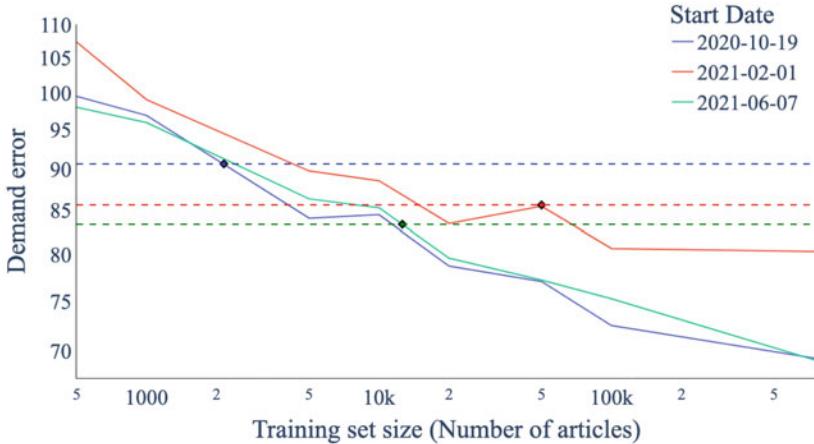
### *Experimental Setup*

- We chose three different forecast start dates distributed over 2021.
- We use 208 weeks of history to forecast 5 weeks for each each start date.

- On each of the three start dates, we use a representative subset of roughly 760K different articles and always use these as the reference point for testing with regards to the Demand Error. In this sense, we will have two layers of generalization present: The first stemming from predicting future demand from time series whose history is present in the training data and the second corresponding to completely unseen parts of the assortment.
- For smaller training sets we ensure that we always have a comparable number of batches per epoch (and hence, comparable training time) by iterating over the same time series several times.
- All other hyperparameters of the training procedure (batch size, cloud instance, learning parameters,...) remain constant and mimic our production setup.
- We included a naive forecast based on the last week's observation in the training horizon to check under which train dataset sizes our model starts to outperform.

The demand error for each train data set size (in terms of the number of time series) as well as different start dates are shown in Fig. 11.11. The naive forecast exhibits the expected behaviour and its performance is naturally constant in the size of the training set. Interestingly, the naive forecaster is already outperformed by training our model on a small fraction of the assortment (between 0.2% and 6.6% for the three start dates under consideration). For two out of the three start dates, the scaling law holds well. Similar scaling laws on public data sets in forecasting with other transformer-based models have not been reported so far (and rather the contrary on publicly available data sets [Borchert et al., 2022, Figure 2]).

Clearly, the above experiment is only a first step in understanding a potential superiority of the transformer-based approach. Our data sets and associated training times are still humble compared to what state-of-the-art NLP-models use, so using more available data (e.g., by going further back in time) presents an obvious opportunity. We believe that this is the first time that evidence for scaling laws for transformers in forecasting was presented.



**Fig. 11.11** Log-log plot of the training-set-size-to-demand-error relationship for three different start dates with constant test set. The dashed lines represent the forecast performance of the naive forecaster (last observation in training horizon)

## RELATED WORK

Our demand model is a customized transformer-based forecasting model (Vaswani et al., 2017). When we launched it in 2019, the current rich landscape of transformer-based forecasting models was not yet available. So, we drew inspiration from advances in deep learning architectures applied to sequential learning problems (Sutskever et al., 2014) and from the success of the transformer architecture in the field of natural language processing (Devlin et al., 2018). In this, our development is similar to other online retailers (Eisenach et al., 2020) who worked on transformer-based time series models concurrently to our work, but prior to the publicly available state of the art. For the latter, remarkable current transformer models include (Li et al., 2019; Lim et al., 2021; Zhou et al., 2020). But, it remains an open question how well these fare against proven state of the art deep learning based models (e.g., Oreshkin et al., 2019; Salinas et al., 2020): We have not yet compared ourselves against these baselines for practical reasons and the introduction of state of the art open-source libraries (Alexandrov et al., 2019) will facilitate this. We

refer to (Benidis et al., 2022) for an overall overview on deep learning based forecasting models.

Note that the approach described here is a point forecast. While clearly not best practice, the (piece-wise) linearity of the profit function used downstream (Li et al., 2020) partially remedies this methodological short-coming. We point to Gouttes et al. (2021) how to turn our model into a flexibly parameterized probabilistic forecasting model. Other generic approaches amend themselves readily to our model such as conformal forecasting (Stankeviciute et al., 2021) or quantile-based approaches (Gasthaus et al., 2019; Kan et al., 2022). We note further that we could handle the intermittency of the demand data more directly (e.g., Jeon & Seong, 2022; Türkmen et al., 2021). This is future work.

Modelling the demand response function as a piece-wise linear function is, to the best of our knowledge, novel. However, we remark that in other contexts, this has been proposed before in forecasting (Gasthaus et al., 2019) and the general theme of modelling the monotonic relationship between price and demand is explored recently (e.g., Loh et al., 2022) although not for deep learning based models.

Finally, we remark that our model (11.1) is an over simplification in the sense that we ignore cross-item relationships or the hierarchical relationship of our data. Recent work for multi-variate forecasting in a deep learning context (e.g., de Bézenac et al., 2020; Rasul et al., 2021; Salinas et al., 2019) and hierarchical forecasting (e.g., Han et al., 2021; Rangapuram et al., 2021; Theodosiou & Kourentzes, 2021) exists, but we leave the application of it for future work.

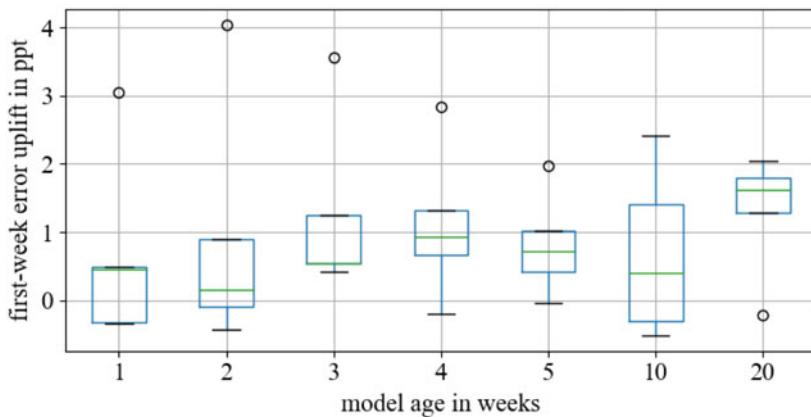
## PRACTICAL CONSIDERATIONS

We have so far described the data and the model assuming a mostly static setting. In reality however, forecasts need to be made available on a weekly schedule in an ever-changing environment. This leads to additional challenges which in turn requires a number of additional artefacts. We summarize this briefly on a high level.

First, we must be able to judge whether our live forecasting accuracy is in line with the accuracy we have in our backtests. For this, we have extensive monitoring metrics that compare incoming live sales data with our forecasts and alarm us, if we observe a deterioration in practice. Second, we must be able to roll out model changes (e.g., adjustments of hyper-parameters, additional covariates or library updates would be small

examples). For this, a deployment pipeline is used with a large suite of test scenarios that cover aspects such as accuracy or training and inference speed in a variety of scenarios. Third, we monitor the input data distribution to scan it for distribution shifts. Note that we could use this for example for sophisticated re-training scheduling schemes, however, experiments reported in Fig. 11.12 have shown that a weekly retraining scheme offers best results.

Finally, we note that the demand forecasting model described here, is only one input of more into a discount optimizer (Li et al., 2020). While our demand forecasts and price elasticities have some value in themselves, the most important artefact are discount decisions. We are only at the start of the journey to understand how forecast accuracy changes attribute to price changes in the complicated interplay between multiple models and (price) decisions that change the environment (and for which backtesting is hence hard). This is particularly involved because our systems evolve dynamically over the course of the seasonal cycle.



**Fig. 11.12** Effect of retraining on accuracy: The plot shows how much accuracy the model loses if not retrained regularly, provided a certain level of randomness introduced by the training process

## CONCLUSION

We have presented the demand forecasting model employed by Zalando SE, one of Europe’s leading fashion online retailer. This global, transformer-based model delivers demand forecasts week in and week out since its initial inception in 2019, thereby predating some of the existing literature on transformer-based forecasting models by at least a year. We showed that this model is competitive against other methods. We also provide a first explanation of the superiority of transformer-based models by testing for scaling laws, as our closed data set contains millions of time series.

The incorporation of price or discount is the main modelling innovation that we presented here and it also presents the largest opportunity for future work. We are in fact most interested in causal or counterfactual forecasts—because these allow the most meaningful downstream decisions. Work on causal forecasting exists (e.g., Melnychuk et al., 2022; Vankadara et al., 2021), but is in its infancy. We believe that much further work is needed and hope that the exposition of our problem presented here will help to inspire such work further.

**Acknowledgements** This article represents the effort of a larger group: Adele Gouttes, Zhen Li, Mateusz Koren, Johannes Stephan, Tofiq Naghibi, Mariia Bulycheva, Matthias Grzeschik, Armin Kekić, Michael Narodovitch, Kashif Rasul, and, Julian Sieber. Each contributed to the manuscript and methodology in unique and impactful ways. We are grateful for their contributions without which we would not have this manuscript (and neither the model running in production at Zalando). It is unfortunate, that, due to restrictions by the publisher, that we cannot acknowledge their contributions more directly via coauthorship as we would have liked and as they would have deserved. In the pre-print version of this manuscript, we properly acknowledge their contributions as coauthors.

## REFERENCES

- Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J. et al. (2019). Gluonts: Probabilistic time series models in Python. *Journal of Machine Learning Research*, 21(116), 1–16.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate*. Paper Presented at 3rd International

- Conference on Learning Representations, ICLR 2015. arXiv preprint. [arXiv:1409.0473](https://arxiv.org/abs/1409.0473)
- Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.-X., Callot, L., & Januschowski, T. (2022, December). Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6), 1–36. ISSN 0360-0300. <https://doi.org/10.1145/3533382>
- Bica, I., Alaa, A. M., Jordon, J., & van der Schaar, M. (2020). *Estimating counterfactual treatment outcomes over time through adversarially balanced representations*. International Conference on Learning Representations (ICLR). arXiv preprint [arXiv:2002.04083](https://arxiv.org/abs/2002.04083)
- Bohlke-Schneider, M., Kapoor, S., & Januschowski, T. (2022). Resilient neural forecasting systems. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*. <https://arxiv.org/abs/2203.08492>
- Borchert, O., Salinas, D., Flunkert, V., Januschowski, T., & Günnemann, S. (2022). *Multi-objective model selection for time series forecasting*. <https://arxiv.org/abs/2202.08485>
- Böse, J.-H., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M., & Wang, Y. (2017, August). Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, 10(12), 1694–1705. ISSN 2150-8097. <https://doi.org/10.14778/3137765.3137775>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901). Curran Associates. <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf>
- de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., & Januschowski, T. (2020, December 6–12). Normalizing Kalman filters for multivariate time series analysis. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
- Eisenach, C., Patel, Y., & Madeka, D. (2020). *MQTransformer: Multi-Horizon forecasts with context dependent and feedback-aware attention*. <https://doi.org/10.48550/arXiv.2009.14799>

- Faloutsos, C., Gasthaus, J., Januschowski, T., & Wang, Y. (2019). Classical and contemporary approaches to big time series forecasting. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19 (pp. 2042–2047). New York, NY, USA. Association for Computing Machinery. ISBN 9781450356435. <https://doi.org/10.1145/3299869.3314033>
- Fildes, R., Ma, S., & Kolassa, S. (2022). Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4), 1283–1318. ISSN 0169-2070. <https://www.sciencedirect.com/science/article/pii/S016920701930192X>. Special Issue: M5 competition.
- Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., & Januschowski, T. (2019, April 16–18). Probabilistic forecasting with spline quantile function RNNs. In K. Chaudhuri & M. Sugiyama (Eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research* (pp. 1901–1910). PMLR. <https://proceedings.mlr.press/v89/gasthaus19a.html>
- Gouttés, A., Rasul, K., Koren, M., Stephan, J., & Naghibi, T. (2021). *Probabilistic time series forecasting with implicit quantile networks*. <https://doi.org/10.48550/arXiv.2107.03743>
- Han, X., Dasgupta, S., & Ghosh, J. (2021, April 13–15). Simultaneously reconciled quantile forecasting of hierarchically related time series. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research* (pp. 190–198). PMLR.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). IEEE.
- Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1), 167–177. ISSN 0169-2070. <https://www.sciencedirect.com/science/article/pii/S0169207019301529>. M4 Competition.
- Januschowski, T., Wang, Y., Torkkola, K., Erkkilä, T., Hasson, H., & Gasthaus, J. (2022). Forecasting with trees. *International Journal of Forecasting*, 38(4), 1473–1481. ISSN 0169-2070. <https://www.sciencedirect.com/science/article/pii/S0169207021001679>. Special Issue: M5 competition.
- Jeon, Y., & Seong, S. (2022). Robust recurrent network model for intermittent time-series forecasting. *International Journal of Forecasting*, 38(4), 1415–1425. ISSN 0169-2070. <https://www.sciencedirect.com/science/article/pii/S0169207021001151>. Special Issue: M5 competition.

- Kan, K., Aubet, F.-X., Januschowski, T., Park, Y., Benidis, K., Ruthotto, L., & Gasthaus, J. (2022, March 28–30). Multivariate quantile function forecaster. In G. Camps-Valls, F. J. R. Ruiz, & I. Valera (Eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research* (pp. 10603–10621). PMLR. <https://proceedings.mlr.press/v151/kan22a.html>
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models*. arXiv:2001.08361.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates. <https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>
- Laptev, N., Yosinski, J., Erran, L. L., & Smyl, S. (2017). Time-series extreme event forecasting with neural networks at Uber. In *International Conference on Machine Learning*, 34, 1–5.
- Lei Ba, J., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization*. arXiv preprint. [arXiv:1607.06450](https://arxiv.org/abs/1607.06450).
- Li, H., Simchi-Levi, D., Sun, R., Wu, M. X., Fux, V., Gellert, T. J., Greiner, T., & Taverna, A. (2020). Large-scale price optimization for an online fashion retailer. *Social Science Research Network*.
- Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., & Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates. <https://proceedings.neurips.cc/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf>
- Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., Delgado, J., Sadoughi, A., Astashonok, Y., Das, P., Balioglu, C., Chakravarty, S., Jha, M., Gautier, P., Arpin, D., Januschowski, T., Flunkert, V., Wang, Y., Gasthaus, J., ... Smola, A. (2020). Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20 (pp. 731–737). New York, NY, USA. Association for Computing Machinery. ISBN 9781450367356.
- Lim, B., Arik, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748–1764.
- Loh, E., Khandelwal, J., Regan, B., & Little, D. A. (2022). Prometheus: An end-to-end machine learning framework for optimizing markdown in online fashion e-commerce. In *Proceedings of the 28th ACM SIGKDD*

- Conference on Knowledge Discovery and Data Mining*, KDD '22 (pp. 3447–3457). New York, NY, USA. Association for Computing Machinery. ISBN 9781450393850. <https://doi.org/10.1145/3534678.3539148>
- Melnichuk, V., Frauen, D., & Feuerriegel, S. (2022). *Causal transformer for estimating counterfactual outcomes*. <https://arxiv.org/abs/2204.07258>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv preprint. [arXiv:1301.3781](https://arxiv.org/abs/1301.3781)
- Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4), 1632–1653. ISSN 0169-2070. <https://www.sciencedirect.com/science/article/pii/S0169207021000558>
- Nair, V., & Hinton, G. E. (2010, June 21–24). Rectified linear units improve restricted Boltzmann machines. In *International Conference on International Conference on Machine Learning* (pp. 807–814). Haifa.
- Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). *N-beats: Neural basis expansion analysis for interpretable time series forecasting*. arXiv preprint [arXiv:1905.10437](https://arxiv.org/abs/1905.10437)
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8024–8035. <http://dblp.uni-trier.de/db/conf/nips/nips2019.html#PaszkeGMLBCKLGA19>
- Pearl, J. (2009). *Causality: Models, reasoning and inference* (2nd ed.). Cambridge University Press.
- Phillips, R. L. (2021). (2nd ed.). Stanford University Press. ISBN 9781503614260. <https://doi.org/10.1515/9781503614260>
- Rangapuram, S. S., Werner, L. D., Benidis, K., Mercado, P., Gasthaus, J., & Januschowski, T. (2021). End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In *Proceedings of the 38th International Conference on Machine Learning* (pp. 8832–8843).
- Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., & Vollgraf, R. (2021). *Multivariate probabilistic time series forecasting via conditioned normalizing flows*. <https://doi.org/10.48550/arXiv.2002.06103>
- Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., & Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank Gaussian copula processes. *Advances in Neural Information Processing Systems*, 32, 6827–6837.
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.

- Stankeviciute, K., Alaa, A. M., & van der Schaar, M. (2021). Conformal time-series forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. Wortman Vaughan (Eds.), *Advances in neural information processing systems* (Vol. 34, pp. 6216–6228). Curran Associates. <https://proceedings.neurips.cc/paper/2021/file/312f1ba2a72318edaaa995a67835fad5-Paper.pdf>
- Sun, F.-K., & Boning, D. S. (2022). *Fredo: Frequency domain-based long-term time series forecasting*. <https://arxiv.org/abs/2205.12301>
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 3104–3112.
- Theodosiou, F., & Kourentzes, N. (2021). *Forecasting with deep temporal hierarchies*. <http://dx.doi.org/10.2139/ssrn.3918315>
- Türkmen, A. C., Januschowski, T., Wang, Y., & Cemgil, A. T. (2021). Forecasting intermittent and sparse time series: A unified probabilistic framework via deep renewal processes. *PLOS One*, 16(11), 1–26. <https://doi.org/10.1371/journal.pone.0259764>
- Vankadara, L. C., Faller, P. M., Hardt, M., Minorics, L., Ghoshdastidar, D., & Janzing, D. (2021). *Causal forecasting: Generalization bounds for autoregressive models*. <https://arxiv.org/abs/2111.09831>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.
- Wen, R., Torkkola, K., Narayanaswamy, B., & Madeka, D. (2017). *A multi-horizon quantile recurrent forecaster*. <https://arxiv.org/abs/1711.11053>
- Yang, S., Eisenach, C., & Madeka, D. (2022). *MQRetNN: Multi-horizon time series forecasting with retrieval augmentation*. <https://arxiv.org/abs/2207.10517>
- Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59(11), 56–65.
- Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2020). *Informer: Beyond efficient transformer for long sequence time-series forecasting*. <https://arxiv.org/abs/2012.07436>



## CHAPTER 12

---

# The Intersection of Machine Learning with Forecasting and Optimisation: Theory and Applications

*Mahdi Abolghasemi*

## INTRODUCTION

Forecasting (predictive analytics) and optimisation (prescriptive analytics) are two widely used analytical techniques in academia and industry.<sup>1</sup> While predictive analytics techniques are used to find the future values of a random variable, prescriptive analytics techniques are used for determining the optimal or near-optimal best decisions given all the available information, i.e., to optimise objective function(s) with respect to some constraints and the existing uncertainty.

<sup>1</sup> In this study, we use forecasting and predictive analytics as interchangeable words. Similarly, (constrained) optimisation, and prescriptive analytics are used as interchangeable words.

---

M. Abolghasemi (✉)  
The University of Queensland, St Lucia, QLD, Australia  
e-mail: [m.abolghasemi@uq.edu.au](mailto:m.abolghasemi@uq.edu.au)

These two methods are the primary tools of decision-making in a vast variety of industries and extremely powerful tools for dealing with uncertainties in operational decision-making. They have been nurtured with decades of research and contributed to each other in several ways. For example, it is a common practice to estimate the parameters of regression models with optimisation models, i.e., by minimising the residuals. On the other hand, parameters of optimisation models are either predicted with some models or assumed to follow a certain distribution. The intersection of prediction, optimisation, and machine learning (ML) gives rise to several interesting problems that lack theoretical and empirical results. By leveraging ML models, we aim to explore two of them including (i) *predict and optimise*, (ii) predicting the decisions of optimisation problems.

In the paradigm of *predict and optimise*, we deal with problems that require both predictions and optimisations for decision-making. This paradigm is useful when the optimisation problem needs input that is not fully observed but needs to be predicted. In this paradigm, we typically predict an unknown variable that will be used as an input parameter in an optimisation problem for determining optimal decisions. This paradigm is of interest from a forecasting perspective as it is concerned with a classic problem in forecasting literature whether forecasting accuracy translates to better decisions in downstream operations and how possibly predictions accuracy can impact the quality of final decisions. While often the objective of the problems that fall into this paradigm is not forecasting but the final decisions, from a forecasting perspective we are interested to know how forecasting accuracy from the first phase translates into decisions in the second phase. The literature suggests that better predictions do not necessarily lead to better decisions but there may be a correlation between them. The nature of the relationship between prediction accuracy and downstream decisions is still an open problem for researchers and practitioners (Abolghasemi et al., 2021).

In the paradigm of *predicting the decisions of optimisation problems*, the goal is to predict the values of decision variables for optimisation problems in real-time. Optimisation problems underpin rich mathematical backgrounds and have found numerous applications in solving real-world problems, e.g., scheduling trains and services, mixing ingredients, and determining facility locations, but their computational complexity may hinder their real-time value and usage in practice, although there has been

an enormous success in speeding them up during the last decade. Nevertheless, any improvement in fast and reliable solving of these problems can be of huge benefit in practice. One common approach to predict decisions is to collect training data offline (often solved by the mathematical model) and then present it to the model in supervised ML models. The availability of training data will not be an issue because one can solve them multiple times offline and collect data for training the ML models. However, there are some issues that must be acted on with care, the most important of which is ignoring the notion of constraints in the ML model that are inherent to the optimisation problem and encountering the danger of violating them. We will focus on the problems where predictive ML models can be used to estimate the decisions of optimisation problems while committing to the constraints or at least trying to abide by them. This paradigm is not investigated in the forecasting literature enough and lacks empirical and theoretical results, in general.

In the following sections, we discuss the latest research and applications concerned with the above-mentioned problems to open up an avenue for operations research researchers and practitioners who are interested to implement them or further exploring them. There are tremendous opportunities that lie in the intersection of these two techniques and using them in conjunction with ML, whether for solving operational problems in practice or for expanding the state-of-the-art in literature. In the following sections, we will explore some of the recent advances in their theory and applications.

## PREDICT AND OPTIMISE

### *Problem Setting*

Forecasting is the basis for many managerial decisions that involve uncertainty. Forecasting per se may not be the end goal but is often a means to an end goal which is making decisions for a problem, whether that is determining the inventory level, scheduling some activities, or production planning.

In the paradigm of *predict and optimise*, the optimisation model takes inputs from the generated predictions by some model and accordingly determines an optimal solution given all the provided inputs. There is always a degree of error associated with these inputs (forecasts), and the optimisation models may accordingly change the prescribed solution with

respect to the input values. However, it is not evident how forecast accuracy will be translated to different decisions in the optimisation phase where we aim to maximise or minimise an objective function and how we can manage the uncertainty in them.

This problem is important from both predictive analytics and prescriptive analytics perspectives. From a predictive analytics perspective, we are interested to know whether higher accuracy of predictions leads to a better decision (Abolghasemi & Esmaeilbeigi, 2021). Generally, forecasters are interested to improve a model that not only predicts accurately on average but it is accurate over the horizon. While some forecasting and decision-making studies argue that forecasting is a separate matter from decisions and each of them should be modelled on their own, others argue that forecasts are not the final goal and should be used for better decision-making purposes; otherwise, there is no real benefit in them (Goodwin, 2009; Guo et al., 2013). From prescriptive analytics, we are interested to develop models that can deal with the uncertainty in the predictions and find the optimal solutions for the entire decision-making horizon.

In a conventional *predict and optimise* problem, prediction and optimisation are typically modelled separately and sequentially. If the predictions are unbiased and accurate, i.e., close to the actual values, then sequential prediction and optimisation should result in (near) optimal decisions if the optimisation problem is modelled and solved to (near) optimality. But this is seldom the case because the prediction and optimisation problems are often complex and more importantly the prediction model learns from a loss function that is typically not aligned with the optimisation model. So, the forecasting model ignores the final decisions in the learning process, i.e., its loss function is independent of the decision optimisation model. As such, the sequential prediction and optimisation will be sub-optimal. The integration of these two steps with ML can be beneficial to achieve a global optimal solution for the final decisions. Different techniques and methods have been proposed for integrating them and improving their performance. While there are no universal solutions for these problems, we explore some approaches that have shown promising results in practice.

### *Methodologies and Applications*

As mentioned before, the conventional approach for *predict and optimise* is to solve them independently, i.e., generate forecasts with the model of

choice and use them as inputs in the optimisation model to determine optimal decisions. Therefore, these approaches are more in the sequential form of ‘predict and then optimise’. While this approach is feasible and may lead to good results, it may not generate the overall optimal solutions. In this section, we discuss how these two phases can be integrated into a single integrated model to improve the results, i.e., forecast accuracy, and decision quality. We advocate the *predict and optimise* rather than *predict then optimise* approach. We look at several different problems of this nature and explore how different techniques can be used to solve them.

One promising method that has been proposed and developed over the years is to use a customised loss function for the forecasting model to embody and learn from the optimisation model. In a seminal work by Elmachtoub and Grigas (2022), authors developed a customised loss function called smart predict and optimise (SPO) where they included the objective and constraints of an optimisation model in the prediction phase. They introduced a decision-error function, i.e., the regret function, that captures the decision errors measured by the difference between the decisions propagated by the predicted values and the actual values if they were known. Since calculating SPO is computationally expensive, they propose a convex surrogate loss function called SPO+, which is time efficient and works well with any polyhedral, convex, and mixed-integer programming (MIP) problems with the linear objective. They discuss three different problems that are common in the realm of *predict and optimise* in real-world: (i) vehicle routing with a focus on the shortest path where the goal is to find the shortest path through a network of cities while the driver needs to go through all cities, each one only one time, starting from the origin and ending in the destination, (ii) inventory optimisation where predicted demand is used as an input to find the optimal lot size or replenishment plan, (iii) portfolio optimisation where the predicted price of stocks are used as inputs in an optimisation model to select a certain number of them, subject to the available budget or risk, and in order to maximise the profit. However, the authors run computational experiments only on the shortest path problem and portfolio optimisation problems and show that the SPO+ is consistent with the least squared and is superior to other conventional loss functions when the model is misspecified. Mandi et al. (2020) looked at the smart predict and optimise for hard combinatorial optimisation problems.

They further explored the SPO and SPO+ in discrete optimisation problems where the objective is not linear. Such a problem is computationally expensive since one needs to solve the optimisation problem sequentially. They relaxed the optimisation problem, i.e., linear problem instead of a MIP problem, and explore how warm-start learning and relaxed optimisation can improve their results. Warm start learning in this context refers to the learning of the predictive model first from the standard mean squared error loss function and then learning from the regret function. Their empirical results on the Knapsack problem and Irish electricity data show that the relaxed optimisation problem achieves similar results to the full discrete optimisation problem. In terms of computational time, warm-start learning speeds up the learning at early stages but does not depict a long-term gain. While SPO shows some promising results on small discrete optimisation problems, they are not tractable on large problems. In an interesting work by Elmachtoub et al. (2020), authors trained decisions by SPO trees loss function which was inspired by SPO and showed that the new decision trees have a higher accuracy and decision quality in comparison with conventional tree-based models. Their model is more interpretable and provides a base for using tree-based models to *predict and optimise* problems.

Another class of methods that are used in *predict and optimise* paradigm is integrated decision-focused models. This approach aims to address the main drawback of the *predict and optimise* problems, i.e., treating these two phases independently (two-stage models) and training them via different loss functions that may not be aligned.

Wilder et al. (2019) proposed an integrated end-to-end model for decision-focused learning where they trained a deep learning model that directly learns the decisions. They focused on two classes of combinatorial optimisation problems including linear programmes and submodular maximisation problems, both of which have many real-world applications, e.g., finding the shortest path with linear programmes, and recommendation systems with submodular maximisation. The challenge for integrating the two phases when the optimisation models are not convex lies in the nature of solving these problems. While prediction models are learned with gradient descent, the optimisation models may be non-differentiable. To remedy this problem, they propose to relax the optimisation problem to a differentiable continuous problem and derive a function that can be used as a continuous proxy for training the integrated model. They empirically test their proposed methodologies on matching, budget allocation,

and recommendation problems all of which have an element of prediction and an element of optimisation. They use neural networks for decision-focused learning and compare them with two-stage approaches. The results show that while the accuracy of two-stage models is often better than the decision-focused ones, the quality of solutions for decision-focused models is significantly better than the two-stage models. One interesting finding of their research is that the correlation between the predicted values of the two-stage neural network with ground truth is lower than the ones for decision-focused models, even though the two-stage models outperform the decision-focused models in terms of accuracy. This shows that the decision-focused models can learn better and account for outliers to obtain a better decision overall.

In decision-focused models, decision loss  $dl(\hat{y}, y)$  can be computed via  $z^*(\hat{y}, y)$ . That is, for a set of  $n$  features and labels  $(x, y)$ , we train a predictive model  $M_\theta$  that minimises the  $dl, \operatorname{argmin}_\theta \sum_{i=1}^n dl(M_\theta(x_i), y_i)$ . In this setting, the predictive model is using decision loss for learning as opposed to conventional forecasting metrics. For example, authors integrate the prediction and optimisation models by learning losses through specific tasks rather than using the surrogate loss function (Shah et al., 2022). They propose locally optimised decision loss (LODL) and argue that their model learns the decisions automatically via their proposed methodology as opposed to other integrated models that use a hand-crafted surrogate function for learning the decisions. Suppose for a given set of features  $x$ , a predictive model  $P_\theta$ , generates forecasts  $\hat{y}$ . The predictions will be used to find the optimal decisions  $z^*$  in the optimisation model  $\operatorname{argmin}(f(z, \hat{y}))$  s.t.  $g_i(z) \leq 0$ , for  $i \in 1, 2, \dots, m$ . In LODL, the authors estimate a new parametric loss function  $LODL_\phi(\hat{y}, y)$  that approximates the  $dl$  behaviour and is convex by nature. So, one can train the model with the new loss function. They empirically show that their model works well on three domains of problems including portfolio optimisation, web advertising, and linear models.

As discussed before, training the predictive model with the loss function of the optimisation model may not be feasible and differentiable, a surrogate loss can be useful in this setting. Many different methods have been proposed for crafting differentiable surrogate loss functions. Mulamba et al. (2021) introduced a new class of loss functions that were inspired by noise contrastive estimation and used inner approximation along with a look-up strategy to speed up the computation. Essentially, they proposed a new loss function that is differentiable and does not

require solving the optimisation model. Based on their work, Mandi et al. (2022) proposed to look at decision-focused models as a learning-to-rank problem that aims to learn the objective function while ranking the solutions in terms of the overall objective. In this setting, different surrogate loss functions can be used for ranking the solutions. The authors used point-wise ranking which looks at each feasible solution independently, pair-wise which looks at the best versus rest, and list-wise ranking which looks at the orders of solutions. The empirical results on the shortest path, energy scheduling, and Bipartite Matching show some promising results although not always superior to other existing methods in the literature.

In a study by Demirović et al. (2019), they looked at the *predict and optimise* problem for the knapsack problem. The knapsack problem is a classic combinatorial optimisation problem that includes prediction and optimisation problems. In Knapsack, the decision maker should select a number of items from a set of  $n$  items each of which has a known weight of  $w$  and profit of  $p$  which may be known or not, such that the total value of items is maximised and the total weight of the items does not exceed a fixed size  $c$ . Knapsack can be formalised as follows:

$$\begin{aligned} & \max \sum_{i=1}^n p_i x_i \\ & \text{subject to : } \sum_{i=1}^n w_i x_i \leq c, \\ & x_i \in \{0, 1\} \end{aligned} \tag{12.1}$$

In this problem, if the profit of an item is not known, then we need to predict it. Given that the prediction will have some associated uncertainty and error, the optimisation model may prescribe a different solution depending on the predictions and errors. In this problem, the end goal is to minimise the error in the optimisation model. One way to do so is to minimise the regret of predictions in the optimisation, which is the difference between the optimal solutions with actual values of  $p$ , and prescribed solutions with predicted values  $\hat{p}$ . This is challenging if the regret function is not differentiable and one may use a surrogate loss function to minimise that. They investigated three approaches including indirect, direct, and semi-direct approaches. The indirect approach is simply predicting and optimising individually and independently, the direct approach uses a

convex surrogate of regret as the loss function and solves the optimisation problem at each time stamp, and the semi-direct approach uses a convex surrogate regret function that uses the parameters of the optimisation model but does not need to solve that repeatedly. For each of these approaches, they use different methods for learning and test their methods on two different datasets, one synthetic data and one energy pricing data. They report while there is limited benefit in including an optimisation model for the synthetic data, there may be more benefit for the energy data which is noisier. For the investigated methods, they state that there is a potential in direct and semi-direct methods and they may outperform the indirect methods when the optimisation model is near convex. The indirect methods may be more appropriate for difficult optimisation problems, and more studies are required to reach a concrete conclusion.

One other interesting approach that has been explored in the arena of integrating forecasting and optimisation is to use the inputs of the forecasting model directly in the optimisation phase for making decisions. In this setting, the forecasting model is eliminated but its inputs are used directly in the optimisation model. Bertsimas and Kallus (2020) proposed a model that goes from data directly to the decision. They develop models for a prescription where there is no constraint and show that we can improve the quality of decisions by leveraging the auxiliary variables that are available to the forecasting models but often are ignored by the decision-making model.

In a different setting to the above-mentioned examples, prediction and optimisation have been explored in stochastic programming problems. Stochastic programming is commonly used for decision-making under uncertainty. If the distribution of inputs is known but the objective cannot be defined and solved analytically, one can use Monte Carlo to draw samples and treat the model as a deterministic model. If the distribution is not known, one can empirically estimate their distribution from samples. The traditional approach is to use sample average approximation (SAA) where the actual distribution of a variable is replaced by an estimation. There are several other methods including robust optimisation, robust SAA, and stochastic optimisation (Bertsimas & Dunn, 2019). ML has been used in various forms for solving stochastic programming problems. For example, Donti et[aut]Zico Kolter, J. al. (2017) proposed end-to-end learning in stochastic optimisation problems. They investigated probabilistic ML models that produce predictions by considering

the objective of the stochastic optimisation models. They intended to use the predictions in the stochastic programming model in a way that leads to the best results for the stochastic programming model but not necessarily the most accurate forecasts. They looked at three problems in energy price forecasting and battery storage, load forecasting and generator scheduling, and a synthetic inventory control problem. The empirical results show that the task-based learning where they have learned the predictions for tasks outperforms the conventional stochastic programming methods. Larsen et al. (2022) also looked at the two-stage stochastic programming problem in a similar setting, but their goal was to rapidly predict the tactical solutions in the second stage as they are computationally demanding. They formulated the problem as a stochastic programming problem and predicted the solutions via ML models. Their empirical results on load planning for transportation show that the predictive accuracy of deep learning models is close to the lower bounds of the stochastic prediction programmes that is calculated based on sample average approximation.

These are some of the main methods and studies in the domain of *predict and optimise*. However, there are other studies that have looked at this problem in different domains and with different techniques. For more details, we refer interested readers to a review paper by Kotary et al. (2012).

### ***Discussion and Future Works***

There is a long-standing debate in forecasting literature that looks at the value of forecasts or the utility of forecasts. That is, what is the final goal of forecasts and how they may be used for decision-making (Abolghasemi & Esmaeilbeigi, 2021). This problem has been investigated from both psychological and judgemental aspects (Goodwin, 2009; Syntetos et al., 2010), as well as mathematical and statistical aspects (Abolghasemi et al., 2021).

Forecasting can be done as a separate task that is independent of decision-making. As such, decision-makers can have their interpretation of forecasts and take action accordingly. One may argue that if the purpose of predictive models is to use them for decision-making, we should incorporate the elements of decision-making in the predictive models directly (Bertsimas & Dunn, 2019; Wilder et al., 2019). But it is not always straightforward to integrate some elements of decisions in the forecasting.

This may be due to the complexity of the decision-making model, e.g., if we have a complex objective function that is not differentiable, or the objectives in the decision-making phase have subjective nature and can not be modelled mathematically. For example, it is not evident how one can embed non-monetary objectives or judgemental elements in the decision-making process and optimise them.

The spectrum of the *predict and optimise* problems is vast, and there is no unique solution for them. Depending on the type of prediction and optimisation problem, the number of constraints in the optimisation model, and the final goal, we may be dealing with convex and non-convex problems which require different approaches. Most of the *predict and optimise* methods have been proposed and tested on optimisation problems with a linear objective. The empirical results on linear programming are promising but generally lacking on broader aspects of the optimisation problems, specially MIP problems (Donti, 2017). The optimisation model may have a discrete and non-differentiable function or complex objective function with complex constraints. This will make the problem difficult and in some scenarios computationally expensive. Extending these methods and verifying them on various constrained optimisation problems is an interesting research avenue with numerous real-world applications.

The computational time is an important barrier in implementing some of the constrained optimisation problems. It would be useful to find fast and reliable solutions to these problems to be able to use them for decision-making in real-time. In the next section, we discuss the idea of predicting the solution of optimisation models with surrogate ML models which can help to speed up the process of solving them but there is a need for more rigorous studies in this domain.

As discussed, one way to integrate decisions in forecasting is to use a surrogate loss function in the model. These surrogates are often estimated by adding regularisation terms to the objective function or relaxing the optimisation problem. It is not trivial how one can build these surrogate functions. The main drawback of this approach is that one needs to design a surrogate loss function for each problem individually. There is no rigorous method or algorithm to design them and it is more of an art that is dependent on the problem at hand. Developing a general-purpose algorithm and guidelines, at least for some types of problems, would be useful. Furthermore, no software or package integrates these two phases of prediction and optimisation. While it may not be easy

to develop general-purpose software that covers all types of prediction and optimisation models, we strongly believe that integrated software for simple cases of prediction and linear optimisation could be beneficial.

Another important problem that is worth investigating is concerned with modelling the uncertainty of forecasts and optimisation. We want to make forecasts more accurate where it matters the most and according to the level of risk one can take, and not just on average. This can be done with a probabilistic forecasting model and integrating it with the optimisation phase. However, there is no rigorous study to look at the distribution of forecast errors and examine how they translate to optimisation costs in downstream operations. While there is empirical evidence that the prediction accuracy will impact the optimisation cost (Abolghasemi & Bean, 2022), it is not evident whether and how the optimisation model will be able to deal with forecast misspecifications. SAA has been used in optimisation models to deal with the underlying uncertainty caused by forecasts; however, this is a passive method to manage the uncertainties and one can account for them in the forecasting phase. Empirical and theoretical analytics will be of tremendous value in this domain.

It will be useful for the community to use the standard benchmarking datasets to develop and evaluate *predict and optimise* methodologies. Newsvendor and Knapsack problem could be a good candidate but that is limited to its form of objective and constraints and thus the methodologies may not be applicable for other *predict and optimise* problems. Several studies have looked at specific *predict and optimise* problems such as optimal power flow (Chatzos et al., 2020), and scheduling (Abolghasemi & Esmaeilbeigi, 2021), but the applications are not limited to these problems and it will be beneficial to develop and evaluate these methods in other areas. Some interesting application is portfolio selection where stocks price should be predicted and a set of them selected for maximum profit similar to M6 forecasting competition, production planning where demand for a product needs to be predicted and then accordingly production needs to be scheduled, staff scheduling where job demands should be predicted and then used in an optimisation model for scheduling staff, just to name a few.

## PREDICTING THE SOLUTIONS OF OPTIMISATION PROBLEMS

### *Problem Setting*

In the “[Introduction](#)” section, we reviewed one important intersection of forecasting and optimisation in a well-known paradigm of ‘predict and optimise’ where both predictions and optimisations were required for making decisions. In this chapter, we look at another intersection of prediction and optimisation where predictive ML models are used for decision-making, instead of constrained optimisation models. We first provide a general introduction to constrained optimisation models before looking at various methodologies and applications of ML in constrained optimisation. Constrained Optimisation problems are concerned with finding the (near) optimal or best solutions for a problem with a set of constraints. Constraint optimisation problems aim to minimise a function  $f$  of variables  $x \in R^n$ , subject to a set of constraints  $C$  that must be satisfied:

$$\min_x f(x)$$

subject to:  $x \in C$ . (12.2)

If a solution  $\eta$  satisfies the constraints, it is called a feasible solution, and if  $f(\eta) \leq f(Z)$ , for all feasible  $Z$ , then it is an optimal solution. These problems have found numerous applications in practice and are celebrated for their strength in finding the best or optimal solutions. Some examples include how to schedule staff or cars for a given number of tasks by minimising the total costs, or how to combine some ingredients to satisfy the requirements of a product while maximising the profit.

The optimisation literature benefits from a rich mathematical background and the community have advanced these methods during the last decade significantly. The constrained optimisation problems may be solvable efficiently or sometimes it may be hard to find a tractable solution for them. If the problem is convex, i.e., the objective function is a convex function and the solutions space is a convex set, then there are efficient methods that can find the optimal solutions for the problem. One common case is linear programming problems, where the objective function and constraints are linear for which we can find their optimal solution efficiently. However, if one of the variables needs to be an integer, we

may not be able to find an efficient or optimal solution. These problems are also called, combinatorial optimisation problems, a particular type of constrained optimisation problems that are characterised by their discrete space and combinatorial nature of solutions. If any of the constraints or the objective function is nonlinear, we call them nonlinear programming again these problems are hard in nature and they may not be tractable.

In general, the intersection of ML and optimisation can be explored from two different perspectives, one being the contribution of optimisation to ML, e.g., finding the optimal decision tree with constrained optimisation models (Bertsimas & Dunn, 2019; He et al., 2014; Khalil et al., 2016), and the other being the contribution of ML to optimisation models, e.g., finding the solution of optimisation with ML models (Abolghasemi et al., 2021; Bertsimas & Dunn, 2019). While this can be further classified, without loss of generality, the focus of this study is on the latter.

### *Methodologies and Applications*

The intersection of ML and optimisation problems is heavily influenced by understanding how ML models can help optimisation models find their solutions faster and in a more systematic way. One such contribution of ML to optimisation is to facilitate finding the solutions of optimisation problems, in particular, to find the branching variables in solving MIP problems via using the ‘Branch and Bound’ algorithm (Bertsimas & Dunn, 2017). Finding the branching variables can significantly improve the speed of solving MIP problems.

‘Branch and Bound’ algorithm seeks the best feasible solutions for MIP optimisation problems by enumerating the possible feasible solutions. After constructing the root node which consists of the entire search space, the algorithm decides which node it is going to explore next. This is based on the internal check where the optimal solutions are calculated, and if there is no benefit in exploring the node, the algorithm will discard it. The branching is done in rather a parsimonious way and although there has been a lot of growth in better choosing these variables and also speeding them up, there are no theoretically guaranteed solutions for them.

Typically branching methods can be classified into two main approaches: (i) Strong Branching (SB) which tests variables at each node and choose the best one that can improve the solution from its

current state and decrease the gap to optimally, (ii) Pseudocost Branching (PB) that mimics the behaviour of SB but speeds up the computation by looking at a smaller number of variables for branching. Many different methods have been proposed for branching but most of them rely on manual feature engineering, heuristic, and ad-hoc methods. ML provides an alternative for a more systematic way of branching. There has been a great interest and a number of studies that explore the application of ML in branching. Various techniques such as supervised classification (Alejandro et al., 2015), regression (He et al., 2014), and ranking (Khalil et al., 2016) methods have been successfully used for this purpose. For example, Bertsimas and Dunn (2017) developed a discrete optimisation model to obtain the optimal branching in decision trees. Decision trees essentially branch on features in a recursive manner to find the best possible solutions. The branching is done using a greedy search where the model selects a feature to find a local optimal. There is no guarantee that the constructed tree will lead to optimal results. They developed a mixed-integer model and tested it on 53 datasets with univariate and multivariate settings. The empirical results showed an improvement of 1–2% for univariate models and 3–5% for multivariate models in comparison with the classification and regression trees algorithm. This accuracy depends on the depth of the tree, and the gain in out-of-sample accuracy is often larger for smaller depths.

Another application of ML in optimisation, as mentioned previously, is to find the solution of optimisation models directly rather than facilitating the process to find the solutions. This is useful because ML models are often significantly faster than the optimisation models, and if we can train an ML model to predict the solution of optimisation problems, we can use them for real-time decision-making. Note that the notion of time is a relative concept when we refer to solving optimisation problems. Traditionally, MIP problems are known to be intractable. Although the algorithms developed by researchers, CPLEX, and GUROBI are significantly faster in comparison with a decade ago, in some cases a million times faster (Bertsimas & Dunn, 2019). Not all MIP problems are computationally expensive but it is not uncommon to find NP-hard problems (there may not be an algorithm that can solve the problem in time that is polynomial in the size of inputs) in this class of problems.

How does the process of using ML for predicting the solution of optimisation models work? One can run the optimisation model to create a training dataset. Then, use the set of inputs including parameters, and

outputs including all decision variables to train a supervised ML model. It will be useful to run the optimisation model with different inputs to generate a versatile set of inputs and outputs for training the ML model. This can help the ML model with better generalisation as it can learn the feasible region of the solutions from larger samples with different inputs. Such a model will train based on mapping inputs to outputs according to the loss function of choice. Multi-class classification (Bertsimas and Dunn, 2019) and multi-output regression (Abolghasemi et al., 2021) have been proposed for this purpose. The choice of the loss function is important for the successful implementation of these algorithms. Ideally, the loss function should be customised according to the objective function and constraints of the optimisation model but this may not be always possible.

The benefit of using supervised ML models is that they are significantly faster in comparison with the optimisation solvers, making them a more desirable choice when real-time decision-makings required. While optimisation models are not always slow, they are still lagging behind most ML algorithms in terms of computational speed. Several problems should be treated with caution when using an ML model for predicting the decisions of constrained optimisation models including (i) committing to constraints, (ii) requiring sample data for learning, and (iii) the ability to generalise.

First, ML models generally are unconstrained optimisation models and one cannot guarantee they will commit to the constraints, whether it be physical or business constraints. Some studies attempt to develop a constrained predictive model to estimate the solutions of optimisation models (Chatzos et al., 2020). Several methods have been proposed to consider the constraints in ML models while learning. Perhaps the most promising approach is to use Lagrangian duality (Fioretto et al., 2020b). The Lagrangian method is a promising method in optimisation to solve complex non-convex problems. For many primal non-convex problems, we can build a dual form with Lagrangian. Suppose an optimisation problem

$$\begin{aligned} & \min f_0(x) \\ & s.t : f_i(x) \leq 0, \quad i = 1, \dots, n. \\ & g_j(x) = 0, \quad j = 1, \dots, m. \end{aligned} \tag{12.3}$$

Then, the Lagrangian relaxation function for this optimisation problem is

$$\min L(x, \lambda) = f_0(x) + \sum_{i=1}^n \lambda_i f_i(x) + \sum_{j=1}^m \lambda_j g_j(x), \quad (12.4)$$

where  $\lambda_i$  and  $\lambda_j$  are called Lagrangian multipliers. The violation-based Lagrangian can be expressed as

$$\min L(x, \lambda) = f_0(x) + \sum_{i=1}^n \lambda_i f_i(x) + \sum_{j=1}^m \lambda_j \max(0, g_j(x)), \quad (12.5)$$

In fact,  $\lambda_i$  and  $\lambda_j$  measure the violation degrees for the constraints.

Fiorotto et al. (2020b) used Lagrangian violation-based relaxation in AC optimal overflow problem in power systems. They developed a deep neural network architecture where they adopted the loss function to account for the physical and engineering constraints. They tested their model on empirical real-world data and showed that their model largely satisfies the constraints and generates reliable solutions significantly faster. Fiorotto et al. (2020a) proposed a Lagrangian dual approach to deal with constraints in deep learning models. They applied their model to energy data and showed that the proposed model is two orders of magnitude better than the traditional deep learning models that use mean squared error as the loss function. Tran et al. (2021) used Lagrangian duality and adopted neural networks to address the problem of fairness, accuracy, and privacy in supervised learning problems where data may be imbalanced and learning algorithms may not have all the information due to privacy concerns for proper learning, thus discriminating against some groups. Chatzos et al. (2020) showed how ML models can be used to approximate the solutions of large-scale optimisation problems in power flow settings. They proposed deep learning models with the Lagrangian duality that can achieve the solutions within 0.01% of the optimality in milliseconds while capturing the constraints with high fidelity. Similarly, Donti et al. (2021) proposed a deep learning architecture called Deep Constraint Completion and Correction (DC3) which enforces feasibility and is differentiable. The empirical results on synthetic data and power flow show near-optimal results while committing to the constraints. While the loss in optimally is marginal, the computational speed makes them a viable choice when dealing with problems that require instant decisions.

The above-mentioned models generally consider the constraints in a soft manner, i.e., we should introduce the constraints and they may still violate the constraints depending on how we set them up. However, they do not learn the constraints. This may limit their benefits as they are unable to intelligently learn the constraints. There are some solutions in the literature that suggested overcoming this limitation. Deep neural networks and graph neural networks are two of the widely used techniques to incorporate the constraints of the optimisation models. For example, Detassis et al. (2021) proposed a deep learning model where the models were able to learn the constraints by supervised ML models. This is different to previously discussed models where constraints were not learned but only considered in implementation. They tested their model on a relatively simple constrained model and showed promising empirical results. However, their results are approximate and not exact in terms of committing to the constraints and obtaining final solutions.

Meta-heuristic algorithms and constraint programming are other approaches that can be used to account for constraints and solve optimisation problems. Although constraint programming can be used for solving combinatorial optimisation problems, we differentiate this application. Constraint programming fundamentally relies on logic programming and graph theory from computer science, but it has also benefited from mathematical programming in developing methodologies. A common application of constraint programming is to use it to find heuristic solutions for MIP problems. Constraint programming works by reducing the solutions space until all the decision variables meet all the constraints and then values are assigned to decision variables. ML techniques have been used in optimising the search process and also predicting their solutions. While constraint programming is a paradigm that is used for solving MIP problems, it is a different problem on its own and its solutions can be predicted with ML. Similar to MIP or other optimisation problems, constraints can be considered in different ways. For example, constraint violations can be considered as penalties in the objective function. This may not guarantee that the solution commits to the constraints but it can facilitate that. Another technique is to hard code the constraints in the ML models by enforcing them to fall between the boundaries. See Popescu et al. (2022), for a comprehensive review of the applications of ML in constraint solving. Meta-heuristic algorithms also have been proposed to solve optimisation problems where problems are interactable, and it is hard to find solutions for them in a reasonable time. Many

different types of them have been developed and used, e.g., genetic algorithm, particle swarm optimisation, Tabu search, and simulated annealing, to name a few. These algorithms are heuristic learning models inspired by physical or natural principles. They can be useful and fast in finding solutions to optimisation problems. ML provides an alternative to these models by leveraging to intelligence in data itself rather than mimicking natural principles as done in most meta-heuristic models.

Second, ML models need a set of training data to learn the process. This means we need to first develop an optimisation model and solve it to obtain the solutions. This way the performance of ML models always depends on the performance of the optimisation model, and if the quality of the solutions for the optimisation model is poor, ML models will perform poorly accordingly. Having said that, if the optimisation model is not solved to the optimality or heuristics are used to approximate their solution, then ML models may be able to learn the process better and outperform them marginally and occasionally but not systematically.

Third, ML models may not be able to generalise, if there is any change in the setting of the optimisation problem. There is no empirical or theoretical evidence to show how much the performance of ML models will change if we change the optimisation model. If there is any change, one needs to run the optimisation model to obtain the inputs and outputs, and retrain the ML model to learn the mapping process again. The problems also exist when we have different input and output lengths, although some solutions have been proposed to deal with this problem (Vinyals et al., 2015). If the parameter values are changed but the model is fixed, then this is no longer an issue. However, it may impact the quality of solutions for the ML models. Depending on the type of problem, the models may or may not change frequently. For example, a network of a blood supply chain in hospitals may not change the business setting, and once the model is developed and trained it may not change anymore or at least for a while. However, if there is a change or a different dynamic in the problem setting, then models are needed to change accordingly and previous surrogate ML models may not be valid anymore. One needs to manually train another model and introduce the changes and new settings.

Reinforcement learning (RL) can be used to help with some of the above-mentioned issues. Generally speaking, RL can be used to learn a policy that maps observations (predictions) to actions (decisions) in an optimization problem. One can combine RL with the supervised

learning problem to improve the performance of the surrogate ML models by enabling them for better generalisation, and also for learning the constraints automatically. To know how we can use this in practice, we first provide the basic principles of the RL systems. Almost all of the RL problems can be formulated via Markov Decision Processes (MDP). We can define MDP with the vector  $M = \langle S, A, R, T, \eta, H \rangle$  where  $S$  is the state space comprising the initial and possible solutions for a problem,  $A$  is the set of actions that an agent can take,  $R$  is the reward function that dictates how the actions will improve or decrease the values for a particular solution,  $T$  is the transition function that governs the transitions from one state to another in a conditional form  $p(s_{t+1}|s_t, a_t)$ ,  $\eta$  is a discount factor which takes values between zero and one whose goal is to encourage for short-term rewards, and  $H$  is the length of actions performed to reach the solution. In mathematical terms, the goal is to maximise the reward,  $\max E[\sum_{t=0}^H \eta^t R(s_t, a_t)]$ . Agents in RL algorithms look for an optimal policy that maximises the reward. This can be done either by (i) maximising the value action function which is the expected reward for a policy given a state and an action, or (ii) directly modelling the agent's policy as a function and optimising it to maximise the reward.

To define RL problems in optimisation, one needs to define the MDP as well as a strategy that the agent will follow to find a policy. The aim is to learn the policy in a systematic way so the model can make decisions that lead to the overall best performance. One particular challenge in RL is defining the reward function which is not always trivial to define or estimate rather it is learned through the actions that the agent takes or some advanced predictive methods. If the experts have a complete understanding of the physical and business constraints, ideally they should be included in the model loss function. Otherwise, RL can be used to learn the process by trial and error.

The policy can be learned from the demonstration as done in imitating learning. The idea of providing a sample behaviour for training the model is also known as imitation learning. In imitation learning, a machine is trained to learn a task after being presented to an expert behaviour (e.g., human or sensors), therefore, learning to map inputs to outputs. Machines can also learn from experiences, where the experience can be defined by an MDP, which shows the agent's state, action, reward, and new state after taking an action. Imitation learning is useful for learning complex tasks as it provides the opportunity to demonstrate the behaviour

to a machine without having the exact policy function in place, simplifying the learning process. It has many applications including robotics, self-driving cars, and games to name a few. In these tasks, the machine is supposed to make a decision and take an action given its current state. This may be doable by an optimisation model. However, the number of scenarios over a horizon often becomes too large, making an optimisation model obsolete. Moreover, it is not trivial what the loss and reward function is and how one can define them, making optimisation models impractical for these problems. As such, often human behaviour is shown to a machine (agent) for enabling them to directly learn from experts. The challenge is the generalisation to unseen states in future because we cannot demonstrate all the behaviours to an agent, but show only one specific behaviour that may not be even optimal. It is useful to combine them with RL and optimisation techniques to optimise the policy. For a review of the current state of the art and future challenges, see Hussein et al. (2017).

In applying RL to an optimisation problem, we define an environment and agents whose aim is to find optimal solutions. The agents and environment interact in a setting where the goal is to maximise the total cumulative agents' rewards with respect to a policy. The policy is often stochastic and the agent may get a reward or penalty depending on the action it takes. RL starts with a random policy and improves this policy over time by learning through trial and error actions. The ecosystem also includes a value function which is the expected return for a particular action, given the current state of the agent. The goal is to build an optimal policy, i.e., an optimal loss function. For this, the agent, on the one hand, may try exploring new states for possible rewards and, on the other hand, repeating the actions that it knows or predicts will return a reward, all with the aim of maximising the total reward. This is known as exploration vs exploitation. All these actions take place in an environment that follows the Markov process, i.e., the future state of the system only depends on the current state and it depends on the older states only through the current state. The agent learns from the experiences and decides on a given MDP. After taking the action and moving into a new state, the agents will update the parameters and keep repeating the process until the end of the horizon. RL can be used to optimise the parameters of the policy. Imitation learning via demonstrating the experts' behaviour can be used to speed up policy learning. The most well-known application of

combining imitating learning and RL is in the game ‘Go’ where a convolutional neural network was used for imitation learning from previous experiences and RL was used during the game for refining the weights and adopting the behaviour (Silver et al., 2016).

RL has been used to solve various optimisation problems. Perhaps the most famous application is Travelling Salesman Problem (TSP). TSP is a combinatorial optimisation problem in which for a set of given cities, a traveller must visit each city exactly once while minimising the total distance. For TSP, we can define an MDP as follows: the state is the cities that the traveller has visited so far with the initial state being the starting point, actions are the city that the traveller can visit, the reward is the negative of the tour length, and the transition function is the chain of nodes until all of the nodes have been visited. Bello et al. (2016) and Dai et al. (2017) are some of the early researchers who proposed to use a graph neural network to sequentially select the nodes(cities) on the graph. The selection policy was learned with RL. This approach proved promising and other studies have adopted similar approaches. See Cappart et al. (2021) and Mazyavkina et al. (2021) for a review of graph neural networks in optimisation and RL application in other typical optimisation problems.

### *Discussion and Future Works*

The methods and applications of ML in predicting the decisions of optimisation problems lack theoretical and empirical results. As discussed various ML models such as deep learning, graph neural network, and tree-based models have been used for predicting the decisions of optimisation problems. There is no consensus on which algorithms may be more useful for which type of optimisation problems. The problems are also largely unexplored in the configuration of the ML models, e.g., what type of neural network architecture with what setting of hyperparameters would be more useful.

We previously discussed the possibility of mapping the inputs to outputs for optimisation models via ML models, just like a typical supervised learning problem. The drawback of these techniques is that they typically ignore the physical and engineering constraints such as capacity constraints if there is any. One interesting research avenue is to use constraint learning techniques to learn the constraints from samples automatically. The current techniques can be used to learn soft

constraints where there is a preference for constraints, or for learning hard constraints such as Boolean constraints (De Raedt et al., 2018). The feasibility of constraints is a big challenge in predicting the solutions of optimisation models.

When using ML to predict the solutions of optimisation, one critical point that we should consider is the scale of outputs and the metrics that are used in surrogate modelling. Surrogate models use another model, often simple, with another evaluation metric, to estimate the solutions of an original problem with a different metric. The evaluation metrics may be completely different, and since the goal of surrogate models is to minimise the error of the original model, it is not enough to evaluate their performance based on their metrics, rather the original metrics should be taken into account.

Another limitation of ML models when predicting the solutions of optimisation problems is their limited capability for generalisation to other similar problems. For example, if a parameter is changed or a problem is applied to a different domain, then one needs to train the model from scratch. However, some of the learning may be transferable to other problems. Transfer learning has been successfully applied in various RL and supervised ML problems (Brys et al., 2015), but to date, there has not been any study in implementing transferring learning to optimisation problems.

One way to improve the performance of ML in predicting the solutions of optimisation models is to use more rigorous feature engineering. In typical ML problems, the inputs have been the same as optimisation inputs, but this can be extended to include other features. For example, we can include statistical features or features related to the structure of the optimisation problems such as the number of constraints, and the number of violations of constraints to improve their predictability (Shen et al., 2023). Khalil et al. (2016) have summarised a set of variables that were used for ranking problems in a supervised ML model to choose the best branching variables.

Last but not the least, the integration of predictive and prescriptive models requires software developers' attention. While there are many open-source software that can be used for either of these purposes, there is no software to predict the solutions of optimisation problems or to integrate prediction and optimisation.

## CONCLUSION

In this study, we explored the intersection of ML with prediction and optimisation. We discussed while prediction and optimisation have benefited from many theoretical advances and are largely applied for solving problems that are concerned with uncertainty, there is great power in integrating these two methods for solving real-world problems more efficiently. The rise of ML methods and computational advances provides us with the best tools to integrate these methods and better manage the uncertainty.

We looked at two main paradigms on the intersection of ML, forecasting, and optimisation, namely *predict and optimise*, and *predicting the decisions of optimisation models*. We investigated several methodological advances and applications of each paradigm and discussed the current problems and future opportunities in research and practice. The research is still in its infancy in these paradigms, and there is a tremendous opportunity for researchers and practitioners to innovate. I hope this study can trigger some ideas for future work in this domain.

## REFERENCES

- Abolghasemi, M., Abbasi, B. & HosseiniFard, Z. (2023). *Machine learning for satisficing operational decision making: A case study in blood supply chain*, *International Journal Forecasting*.
- Abolghasemi, M., & Bean, R. (2022). *How to predict and optimise with asymmetric error metrics*. GitHub, Preprint.
- Abolghasemi, M., & Esmaeilbeigi, R. (2021). *State-of-the-art predictive and prescriptive analytics for IEEE CIS 3rd Technical Challenge*. arXiv preprint [arXiv:2112.03595](https://arxiv.org/abs/2112.03595)
- Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2021). *Neural combinatorial optimization with reinforcement learning*. arXiv preprint [arXiv:1611.09940](https://arxiv.org/abs/1611.09940)
- Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106(7), 1039–1082.
- Bertsimas, D., & Dunn, J. (2019). *Machine learning under a modern optimization lens*. Dynamic Ideas LLC.
- Bertsimas, D., & Kallus, N. (2020, March). From predictive to prescriptive analytics. *Management Science*, 66(3), 1025–1044.
- Brys, T., Harutyunyan, A., Taylor, M. E., & Nowe, A. (2015). Policy transfer using reward shaping. In *AAMAS* (pp. 181–188).

- Cappart, Q., Chetelat, D., Khalil, E., Lodi, A., Morris, C., & Veličković, P. (2021). Combinatorial optimization and reasoning with graph neural networks. arXiv preprint [arXiv:2102.09544](https://arxiv.org/abs/2102.09544)
- Chatzox, M., Fioretto, F., Mak, T. W. K., & Van Hentenryck, P. (2020). *High-fidelity machine learning approximations of large-scale optimal power flow*. arXiv preprint [arXiv:2006.16356](https://arxiv.org/abs/2006.16356)
- Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30.
- De Raedt, L., Passerini, A., & Teso, S. (2018). Learning constraints from examples. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1).
- Demirović, E., Stuckey, P. J., Bailey, J., Chan, J., Leckie, C., Ramamohanarao, K., & Guns, T. (2019). An investigation into prediction+ optimisation for the knapsack problem. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (pp. 241–257). Springer.
- Detassis, F., Lombardi, M., & Milano, M. (2021). Teaching the old dog new tricks: Supervised learning with constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 5, pp. 3742–3749).
- Donti, P., Amos, B., & Zico Kolter, J. (2017). Task-based end-to-end model learning in stochastic optimization. In *Advances in neural information processing systems*, 30.
- Donti, P. L., Rolnick, D., & Zico Kolter, J. (2021). *Dc3: A learning method for optimization with hard constraints*. arXiv preprint [arXiv:2104.12225](https://arxiv.org/abs/2104.12225)
- Elmachtoub, A., Cheuk Nam Liang, J., & McNellis, R. (2020). Decision trees for decision-making under the predict-then-optimize framework. In *International Conference on Machine Learning* (pp. 2858–2867). PMLR.
- Elmachtoub, A. N., & Grigas, P. (2022). Smart “predict, then optimize”. *Management Science*, 68(1), 9–26.
- Fioretto, F., Van Hentenryck, P., Mak, T. W. K., Tran, C., Baldo, F., & Lombardi, M. (2020a). Lagrangian duality for constrained deep learning. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 118–135). Springer.
- Fioretto, F., Van Hentenryck, P., Mak, T. W. K., Tran, C., Baldo, F., & Lombardi, M. (2020b). A lagrangian dual framework for deep neural networks with constraints optimization. In *European conference on machine learning and principles and practice of knowledge discovery in databases (ECML-PKDD)*. Vol. 12461. *Lecture Notes in Computer Science* (pp. 118–135). Springer.
- Goodwin, P. (2009). Common sense and hard decision analysis: Why might they conflict? *Management Decision*, 47(3), 427–440.

- Guo, Z. X., Wong, W. K., & Li, M. (2013). A multivariate intelligent decisionmaking model for retail sales forecasting. *Decision Support Systems*, 55(1), 247–255.
- He, H., Daume III, H., & Eisner, J. M. (2014). Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27.
- Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017, April). *Imitation learning: A survey of learning methods*. *ACM Computing Surveys*, 50(2), 1–35.
- Khalil, E., Le Bodic, P., Song, L., Nemhauser, G., & Dilkina, B. (2016, February). Learning to branch in mixed integer programming. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30, No. 1).
- Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). *End-to-end constrained optimization learning: A survey*. arXiv preprint [arXiv:2103.16378](https://arxiv.org/abs/2103.16378)
- Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., & Lodi, A. (2022). Predicting tactical solutions to operational planning problems under imperfect information. *Informs Journal on Computing*, 34(1), 227–242.
- Mandi, J., Bucarey, V., Tchomba, M. M. K., & Guns, T. (2022). Decision-focused learning: Through the lens of learning to rank. In *International conference on machine learning* (pp. 14935–14947). PMLR.
- Mandi, J., Stuckey, P. J., Guns, T., et al. (2020). Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 2, pp. 1603–1610).
- Marcos Alvarez, A., Wehenkel, L., & Louveaux, Q. (2015). *Machine learning to balance the load in parallel branch-and-bound*.
- Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134, 105400.
- Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Bucarey, V., & Guns, T. (2021). Contrastive losses and solution caching for predict-and-optimize. In *Proceedings of the thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*.
- Popescu, A., Polat-Erdeniz, S., Felfernig, A., Uta, M., Atas, M., Le, V-M., Pilsl, K., Enzelsberger, M., & Tran, T. N. T. (2022, February). An overview of machine learning techniques in constraint solving. *Journal of Intelligent Information Systems*, 58(1), 91–118.
- Shah, S., Wilder, B., Perrault, A., & Tambe, M. (2022). Learning (local) surrogate loss functions for predict-then-optimize problems. arXiv preprint [arXiv: 2203.16067](https://arxiv.org/abs/2203.16067).
- Shen, Y., Sun, Y., Li, X., Eberhard, A., & Ernst, A. (2023). Adaptive solution prediction for combinatorial optimization. *European Journal of Operational Research*, 309(3), 1392–1408.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman,

- S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016, January). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.
- Syntetos, A. A., Nikolopoulos, K., & Boylan, J. E. (2010). Judging the judges through accuracy-implication metrics: The case of inventory forecasting. *International Journal of Forecasting*, 26(1), 134–143.
- Tran, C., Fioretto, F., & Van Hentenryck, P. (2021, May). Differentially private and fair deep learning: A lagrangian dual approach. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 11, pp. 9932–9939).
- Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, 28.
- Wilder, B., Dilkina, B., & Tambe, M. (2019, July). Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 1, pp. 1658–1665).



## CHAPTER 13

---

# Enhanced Forecasting with LSTVAR-ANN Hybrid Model: Application in Monetary Policy and Inflation Forecasting

*Michał Chojnowski*

## INTRODUCTION

Neural networks' applications in macroeconomic modelling are still an uncharted territory in the economic literature. The main obstacle to overcome in this field is a relatively small dataset size from which an artificial neural network (ANN) can learn. In general, in the forecasting literature, the “classical” time-series methods tend to outperform machine learning algorithms in forecasting capabilities for small samples (Cerqueira et al., 2022; Makridakis et al., 2018). There are a few articles though, which shows that neural networks can compete with traditional time-series models in GDP forecasting (Tkacz, 2001) and inflation forecasting (Arnerić & Šestanović, 2021; Choudhary & Haider, 2012; Moshiri &

---

M. Chojnowski (✉)  
Warsaw School of Economics, Warszawa, Poland  
e-mail: [michael.chojnowski@gmail.com](mailto:michael.chojnowski@gmail.com)

Cameron, 2000). Moreover, the forecast combination of linear and non-linear for inflation and interest rate brings improvement in accuracy (Choudhary & Haider, 2012; Szafranek, 2019). ANN was also found useful in interest rate forecasting due to nonlinearities in the Taylor rule (Hinterlang, 2020). The literature on ANN applications in monetary policy is limited. Most of the articles focus on an analysis of Central Bank communication to predict interest rate changes in the short run (Gorodnichenko et al., 2021; Hinterlang, 2020).

Internet data analysis literature on the other hand has expanded in the last decade rapidly. It focuses on the impact and relations between the vast dataset and market fundamentals. One of many information easily available for researches is Google Trends. Google Trends has been used in many economic fields such as unemployment (Askitas & Zimmermann, 2009; Barreira et al., 2013; D'Amuri & Marcucci, 2017; Naccarato et al., 2018; Nagao et al., 2019), stock market and portfolio management (Kropiński & Anholcer, 2022; Moat et al., 2013; Petropoulos et al., 2022; Preis et al., 2013), GDP and inflation (Bantis et al., 2022; Bicchali & Raja, 2019; Bleher & Dimpfl, 2022), commodity prices (Salisu & Adewuyi, 2020; Yu et al., 2019), housing markets (Coble & Pincheira, 2017; Dietzel, 2016) or economic uncertainty measurement (Algaba et al., 2020; Baker et al., 2016; Sorić & Lolić, 2017).

Another developing area of the literature is model combination called hybrid models. One of the popular approaches is the ARIMA-ANN model, where ANN predicts residuals of the ARIMA model (Peirano & Kristjanpoller, 2021; Wang et al., 2013; Zhang, 2003). The hybrid model assumes that the ARIMA part captures the linear dependencies, whereas ANN can capture the remaining non-linear ones. A wavelet transformation of the input data is also used often in hydro- and meteo-forecasting literature (Pramanik & Singh, 2011; Shafeai et al., 2016), as well as ensemble empirical mode decomposition EEMD-ANN hybrid model (Qiao-Feng et al., 2018). EEMD-ANN model was also used in predicting COVID-19 cases (Hasan, 2020).

Although the direction in which neural network is taking within the macroeconomic literature looks promising, there are still a few hindrances to overcome. The critique of the ANN forecasts literature concerns a limited benchmark model selection. Naïve forecast and random walk are too weak to be used as a benchmark in some cases. Other time series models such as Exponential Smoothing, ARIMA or VAR are much

stronger candidates. Hence, including those models as benchmarks brings more solid proof of ANN forecast out-performance.

The early literature has implemented Google Trend searches “directly” into the linear models (Askitas & Zimmermann, 2009; Choi & Varian, 2012; Nick & Shanbhogue, 2011). The implementation of good indicators has increased the accuracy of the forecast. However, those models open up a possibility of spurious correlations, in which several phrase queries in the Google search browser directly affect the economic fundamentals.

A recent answer to the small sample problem includes the cooperation of theoretical economic models with ANN. In recent years, articles combining the DSGE models with ANN have been published. The DSGE model is used to generate a training dataset for a neural network (Stempel & Zahner, 2022). With this approach, neural network has enough data to learn the variable-relationship patterns from the theoretical model. Then, a neural network is validated on a real dataset. This approach has been used in evaluating the monetary policy of EBC in Northern and Southern European countries (Stempel & Zahner, 2022).

The author identified a niche in the literature which models the monetary policy using ANN. Part of the monetary policy literature focuses on asymmetric effects on prices (Bolboaca & Fischer, 2019; Weise, 1999). Those articles include LSTVAR model, in which a transition function determines the regime in which economy operates. Each regime is a linear combination of two autoregressive processes.

Considering uncertain macroeconomic forecasting performance of ANN over “classical” models, the author decided to approach ANN application from a different angle. Following the phrase “if you cannot beat them, join them”, the author explores the method in which transition function in LSTVAR model is being enhanced by ANN.

The monetary policy model used in this research is based on Peersman and Smets VAR model (Peersman & Smets, 2001). Additionally, the author imposes an existence of economic regimes with different autoregressive processes in each of regimes, thus leveraging the LSTVAR model (Chojnowski, 2022; Gefang & Strachan, 2009). In this research, the author has chosen a customers’ perception towards their financial and economic well-being as a transition variable. ANN provides more insights into the regimes’ nature and dynamics of endogenous variables within those regions. There is no literature on the LSTVAR-ANN hybrid model known to the author.

Customer confidence is believed to be build based on three main topics: prices, disposable income and employment (Drozdowicz-Bieć, 2011). The author imposes an assumption that with higher disposable income consumers have more money to allocate, either on consumption or on savings. Additionally, the author has considered the fourth group—the capital market. The capital market should enclose the area of economic well-being, which is part of the Customer Confidence Index of Michigan University questionnaire. Moreover, topic related to the consumption and inflation is similar, and hence, the author decided to gather phrase from those two topic groups together. Hence, in this research, four Google Trends topics are considered: prices, savings, capital market and employment. From those topics, the author is extracting unobserved factors, which are further passed into the transition function.

Unobserved factors can be extracted from Google Trends using several methods. Principal Component Analysis (PCA) is one of the most popular as it enables researchers to reduce the dimension of the input (Chojnowski & Dybka, 2017; Szafranek, 2019). Other shrinkage methods are elastic nets, LASSO regression and ridged regression (Bantis et al., 2022). Additionally, neural networks were used as a factor extraction method as well (Singhania & Kundu, 2020). In this chapter, the author considers two methods: PCA and neural network. For “PCA” input LSTVAR-ANN model, unobserved factors are calculated beforehand and then fed into the neural network. For “neural network” input, the ANN part of the model is extended with “aggregation” layers.

Thus, LSTVAR-ANN is a comprehensive model that describes time series’ dynamics and the environment in which they operate. ANN can model impact and relations between observable variables, unobserved hidden factors and transition functions. Therefore, an advantage of the LSTVAR-ANN hybrid model lies in the explanatory power of a transition variable and economy dynamics.

This chapter presents the capabilities of monetary policy analysis and forecasting using LSTVAR-ANN hybrid model. The author focuses on three main parts: analysis of unobserved factors impacting indirectly the inflation forecasting, impact of monetary policy on inflation under customer confidence regimes and point forecast of US inflation.

The chapter follows the structure: firstly, the author presents the technicalities of the LSTVAR-ANN model to the reader. Then, the set-up of the monetary policy model is presented with dataset used. Afterwards, the results of the LSTVAR-ANN model are discussed.

## LSTVAR-ANN HYBRID MODEL

This subchapter presents the LSTVAR-ANN hybrid model method. Firstly, the author introduces the reader to the LSTVAR model and a role of the transition function. In LSTVAR model, there exist multiple autoregressive processes. The final observed outcome is the linear combination of autoregressive processes results. The transition function defines the coefficients of such combination. In this chapter, the author shows how ANN can be implemented into the LSTVAR model as a transition function. Afterwards, two optimisation methods of the LSTVAR-ANN hybrid model are discussed: with and without supervision on a transition variable. In this research, the author has chosen customer confidence as a variable for supervision. The author discusses if the supervision of customer confidence is necessary to obtain the Business Cycle behaviour. Finally, the author shows the method to represent impulse response function (IRF) under different value of transition variable: 3D-IRF. “Classical” IRF shows how response variable (y-axis) changes over time (x-axis) when impulse from another variable is recorded. 3D-IRF adds another dimension—transition function value. Then, 3D-IRF shows how response variable (z-axis) changes over time (x-axis) under transition function value (y-axis) when impulse from another variable is recorded. Thus, 3D-IRF function shows how dynamics of the model changes with change of transition function.

### *LSTVAR Model*

LSTVAR model was well described by Timo Teräsvirta and Yukai Yang (Teräsvirta et al., 2014). Let us define VAR(k) model with series of  $p$  endogenous variables ( $y$ ) and exogenous variables ( $d$ ) as follows:

$$\begin{aligned} y_t &= A'_1 y_{t-1} + A'_2 y_{t-2} + \cdots + A'_k y_{t-k} + \Phi' d_t + \varepsilon_t \\ &= F' x_t + \varepsilon_t \end{aligned}$$

where  $F = (A'_1, \dots, A'_k, \Phi')$  and  $x_t = (y'_{t-1}, \dots, y'_{t-k}, d'_t)$ . Error vector  $\varepsilon_t$  is a white noise with mean zero and positive define covariance matrix  $\Omega$ .

Then, a smooth transition VAR model takes the following form:

$$y_t = F' x_t = \left( \sum_{i=1}^m (G_t^{i-1} - G_t^i) F_i' \right) x_t + \varepsilon_t, \quad (13.1)$$

where  $m$  is a number of regimes and  $F_i = (A_{i1}', \dots, A_{ik}', \Phi_i')$ .  $G_t^i$  contains on its diagonal transition function value of the transition from regime  $i-1$  to  $i$ , where  $G_t^0 = I_p$  and  $G_t^m = 0$ .

This research model is simplified in two ways. Firstly, let us assume that only two pure regimes exist: high and low. Hence, the number of regimes can be set to 2 ( $m = 2$ ). From Eq. 13.1 model then takes a following form (Teräsvirta et al., 2014):

$$y_t = ((I_p - G_t^1)F_1' + G_t^1 F_2')x_t + \varepsilon_t,$$

The other assumption is put on matrix  $G_t$ . If the transition function is the same for all of the variables, then  $G_t$  can be written as the multiplication of a transition function  $g$  (which returns a scalar) and an identity matrix  $I_p$ . Therefore:

$$y_t = ((I_p - g(I_p))F_1' + gI_p F_2')x_t + \varepsilon_t. \quad (13.2)$$

With simple arithmetics, we can rewrite an Eq. 13.2 for easier computation as well as the easier implementation of ANN in the following way:

$$y_t = I_p(F_1' + (F_2' - F_1'))x_t + \varepsilon_t = F_1'x_t + (F_2' - F_1')x_t. \quad (13.3)$$

The model presented in Eq. 13.3 consists of two components: linear one ( $F_1'x_t$ ), describing regime 1, and non-linear part ( $(F_2' - F_1')x_t$ ) describing differences between regime 1 and 2. Although interpreting the results in this form is more challenging, it is much easier computational-wise. Instead of 2 non-linear components as in Eq. 13.2, we reduce the number of non-linear parts to 1.

Transition function values LSTVAR model parameters can be estimated using the grid search method on the set of transition function parameters. Henceforth, another constraint on the feasibility of the results can be assessed, e.g. stability of both pure regimes (Chojnowski, 2022).

In his previous research, the author has used the Customer Confidence Index of Michigan University as a transition variable as it is believed to resemble market sentiments well (Bantis et al., 2022).

### *LSTAR-ANN Hybrid Model*

In the model of Teräsvirta and Yukai, transition function  $g$  is assumed to be a logistic function. Hence, it can take a single variable as an input. In this chapter, the author relaxes this assumption by allowing a reasonably large amount of exogenous inputs for the transition function.

Let us consider a transition function  $\bar{g}$  equivalent to the function  $g$  from Eq. 13.3.  $\bar{g}$  is formulated by an ANN, such that it takes an input set  $I_t$  and returns a single value within interval  $[0, 1]$ :

$$\bar{g} : I_t \rightarrow [0, 1]$$

The simplest example of  $\bar{g}$  would be a single-input network with no hidden layers and a sigmoid activation function. Such architecture would correspond to the logistic transition function  $g$ .

Let us express an LSTVAR model as a neural network  $h$ . The matrices  $F_1$  and  $F_2 - F_1$  can be estimated with ANN using the linear activation function. Hence, the network would require an input set of the previous realisation of vector  $y_t$  (denoted as  $x_t$ ) and a function  $\bar{g}$ . The non-linear component requires multiplying the input vector by the results of transition function  $\bar{g}$ . Thus, architecture of  $h$  follows the list below:

1. Estimate transition function  $\bar{g}$  value,
2. Take input vector  $x_t$  and multiplying it by results of  $\bar{g}$ ,
3. Concatenate input vector  $x_t$  and results of point 1, and predict  $y_t$  using linear activation function.

Hence,  $h$  is a function, which takes  $\bar{g}$  and  $x_t$  as input and  $y_t$  as output:

$$h(x_t, \bar{g}) : [x_t, \bar{g}] \rightarrow y_t.$$

However, two neural networks can be easily merged. Let us label a complete LSTVAR-ANN model as  $\bar{h}$ . Then,  $\bar{h}$  takes two input sets— $I_t$  and  $x_t$  and returns  $y_t$ :

$$\bar{h}(I_t, x_t) : [I_t, x_t] \rightarrow y_t.$$

There is an additional aspect to the model. Recalling transition function  $\bar{g}$ , let us assume that  $\bar{g}$  aggregates input set to the single variable  $\xi_t$  and with its last layer, it rescales  $\xi_t$  into interval  $[0, 1]$ . Then, we open

up the possibility that a neural network can train itself to minimise the error between its aggregation  $\xi_t$  and external proxy  $\tilde{\xi}_t$ . Let's call that architecture "sentiment-supervised" ( $\bar{g}_s$ ) and previously discussed  $\bar{g}$  as sentiment-unsupervised. Analogically, if LSTVAR-ANN is using  $\bar{g}_s$  as a transition function, let us label it as sentiment-supervised LSTVAR-ANN ( $\bar{h}_s$ ) and if it is using  $\bar{g}$ —sentiment-unsupervised LSTVAR-ANN. Henceforth, a sentiment-supervised LSTVAR-ANN takes two sets as inputs— $I_t$  and  $x_t$  and returns two outputs— $y_t$  and  $\xi_t$ :

$$\bar{h}_s(I_t, x_t) : [I_t, x_t] \rightarrow [y_t, \xi_t].$$

The sentiment-supervised method brings an advantage in computation time. Suppose we know a transition variable which defines LSTVAR regimes. In that case, sentiment-supervised architecture can adjust the aggregation process of  $I_t$  to match a proxy and select the most influential factor. If such variable is unavailable, then sentiment-unsupervised LSTVAR-ANN architecture establishes transition function values with the information given.

#### *Research Architecture Specification*

In this research, the author has implemented two additional aspects. First, for sentiment-supervised architecture, the author assumes that a layer prior to aggregate  $\xi_t$  would represent the unobserved factors, which affects the customer confidence index, similarly, as in Dynamic Factor Model (Giannone et al., 2008). Therefore, set of Google Trends has been carefully selected to comply with market sentiments literature.

Secondly, these factors are connected to Google Trends' query topic only. Hence, the author knows if unobserved factors are related to, e.g. prices or unemployment. Therefore, the final architecture of  $\bar{g}_s$  works as follows:

1. Group Google Search queries into topics
2. Aggregate  $i$ -th group into  $N_i$  factors with a neural network  $\bar{g}_{i,s}$
3. Find relation between unobserved factors and aggregate  $\xi_t$
4. Transform aggregates  $\xi_t$  into interval  $[0, 1]$

For the reader's curiosity, the author attaches the sentiment-supervised LSTVAR-ANN architecture sketch in the Appendix (Fig. 13.9).

Another approach assumes aggregating the Google Trends outside of the neural network and uses aggregates as input for transition function layer. In this research, the author uses PCA method for dimension reduction.

### ***3D Impulse Response Function***

The “classical” IRF shows the response of the variable  $i$  if shock in variable  $j$  is observed. The IRF values are obtained by inverting the  $VAR(1)$  model into  $MVA(\infty)$  model. It is worth mentioning that any VAR of order  $k$  can be expressed as a VAR model of order 1.

Let's rewrite the LSTVAR model from Eq. 13.3:

$$y_t = I_p Ax_t + \varepsilon_t, \quad (13.4)$$

where

$$A = F_1' + g(F_2' - F_1').$$

The values of matrices  $F_1$  and  $F_2$  are derived from weights of linear activation functions.

If value of the transition function  $g$  is known, the calculation of the IRF is straightforward. Hence, the author proposed a following method to calculate the 3D-IRF:

1. Create a placeholder for IRF values
2. Loop over the interval  $[0, 1]$  with an iterator  $i$
3. Set value of the transition function to  $i$
4. Calculate IRF
5. Append the placeholder by current loop IRF values

The result of the algorithm is a 2-dimensional matrix with 3D-IRF values. By plotting the matrix using, e.g. heatmap, the reader can easily identify the changes in the dynamics of the model and inter-variable interactions. In this research, when the transition function inputs are market sentiments, the reader can follow how effective would be increase of interest rate under different market sentiment values.

## MONETARY POLICY MODEL WITH REGIMES

Analysis of the monetary policy in the United States is based on the approach proposed by Peersman and Smets (2001). Let us consider a VAR model with the following variables: production output ( $y_t$ ), consumer price index ( $p_t$ ), interest rate ( $i_t$ ) and the exchange rate ( $s_t$ ).

The regimes are identified based on the Customer Confidence Index as an external proxy ( $\xi_t$ ). There are two pure regimes: the high confidence and the low confidence value regimes. The author uses an identity matrix as a structural matrix for simplicity. This simplification affects IRF values only. The author uses autoregression of order 3 in this research. Hence, the final model with Google Trends dataset ( $I_t$ ) looks as follows:

$$\begin{aligned} Y_t &= [y_t, p_t, i_t, s_t], \\ X_t &= [Y_{t-1}, Y_{t-2}, Y_{t-3}], \\ Y_t &= F_1' X_t + g(\cdot)(F_2' - F_1') X_t + \varepsilon_t, \end{aligned}$$

where:

$$g(\cdot) = \begin{cases} \bar{g}(I_t) & \text{for sentiment-unsupervised} \\ \bar{g}_s(I_t, \tilde{\xi}_t) & \text{sentiment-supervised} \end{cases},$$

$$\varepsilon \sim N(0, \sigma_\varepsilon).$$

## DATA

In this research, models were estimated based on monthly data. If necessary, variables are seasonally and calendar adjusted.

Although the data for the United States have been available for a very long period of time, the author decided to start our analysis in 2004 due to Google Trends availability. The macroeconomic data for the monetary model are obtained via Federal Reserve Economic Data (FRED): production [INDPRO], CPI [CPILFESL], effective federal funds rate [DFF] and real effective exchange rate [RNUSBIS]. The Michigan Consumer Index is used as an economic sentiment indicator.

Google Trends dataset starts in January 2004 and is of monthly frequency. All Google Trends are seasonally adjusted. The author has chosen 51 phrases and categorised them into four topic groups: prices, savings, capital and employment. A list of used queries can be found in Appendix (Table 13.5)

## RESULTS

In this section, the author presents the analysis of the LSTVAR-ANN hybrid model. If not stated otherwise, results are provided for the pre-COVID-19 period (which ends in December 2019). As of writing this chapter, COVID-19 pandemic is still present with the ongoing war in Eastern Europe. Due to the current extreme events, the author cannot distinguish between model uncertainty and the events' impact mentioned above.

The author also tests if dimension reduction done beforehand affects model performance. The author has used the PCA method on the Google Trends dataset to extract possible unobserved factors. All factors with variance coverage over 5% were taken as input variables.

Hence, in sections “[Transition Function Output](#)” and “[Forecasting](#)”, author is comparing the results of four models: sentiment supervised or sentiment unsupervised and Google Trends or PCA input. For other sections, the results of the sentiment-supervised LSTVAR-ANN model are discussed.

The section is structured in the following way: firstly, the author extracts and discusses unobserved factors from the sentiment-supervised LSTVAR-ANN model. Next, the importance of unobserved factors and Google Trends phrases on CPI predictions is presented. Afterwards, the author discusses regime occurrence based on transition function values. The results are compared across four estimation types of LSTVAR-ANN. In section “[Response of CPI on Interest Rate Shock](#)”, the author presents the results of 3D-IRF of monetary policy shock on CPI. Lastly, the author presents the forecasting accuracy of the LSTVAR-ANN model compared to 5 benchmark models.

### *Unobserved Factors Extraction*

Sentiment-supervised LSTVAR-ANN model with Google Trends input enables quantitative analysis of unobserved factors and identification of driving Google Trends queries. The number of unobserved factors has to be stated beforehand in the architecture. The author has chosen the same number of factors as the number of PCA-relevant components per topic. The author calculated the importance of each query on each factor and presented queries with a relative importance of 0.7 or higher. Based on the preselected list of queries (see Appendix, Table [13.5](#)), the author

interprets extracted factors in Table 13.1. The ID of the factor consists of the Google Trends origin topic and an identification number.

$Prices_0$  represents the perception of the gasoline market. Present and lagged values of oil prices dominate the impact of this factor.  $Prices_1$  comes from the “SoCal” query, which represents SoCalGas company. Hence,  $prices_1$  represents the gas industry. However, SoCal has several meanings (e.g. the name of the Southern California region), which might add a misleading noise.  $Prices_2$  main components describe the car-related market. Surprisingly, other queries (such as retail markets and other economy segment prices) were not selected as the most impactful. It

**Table 13.1** Top drivers for unobserved factors

Description	Factor ID	Google Trends queries
Oil market	$prices_0$	oil prices [0, -1, -2], gas prices [-1]
Gas market	$prices_1$	SoCal [0, -1, -2]
Car market	$prices_2$	SoCal [0, -1, -2], car prices [0, -1], gas prices [0, -1]
Financial instruments	$savings_0$	returns [-1, -2], mutual funds [-2], Wells Fargo [-1]
Credit cards	$savings_1$	credit card [0, -1], Citibank [-1], Wells Fargo [-2]
Investment	$capital_0$	bearish market [0, -1], IPO [0], sp500 [0], estate agents [0, -2]
Bear market	$capital_1$	bearish market [-1]
Real estate market	$capital_2$	estate agents [0, -1, -2], real estate [0, -1, 2], bearish market [-2]
IPO occurrences	$capital_3$	bearish market [-1, -2], IPO [0]
Stock market performance	$capital_4$	sp500 [0, -1, -2]
Current unemployment	$employment_0$	unemployment [0], Berkshire Hathaway [0]
Career seek	$employment_1$	State Farm [0], Berkshire Hathaway [0, -1], career [-1]
Job market slowdown	$employment_2$	unemployment [0, -1, 2], Berkshire Hathaway [0, -1, -2], State Farm [0], bailout [-2]
Insurance	$employment_3$	State Farm

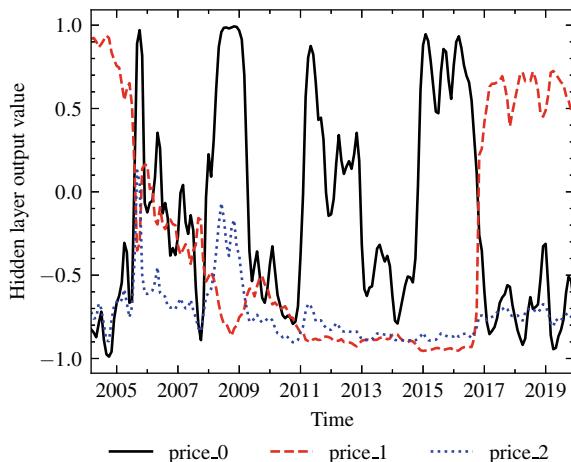
*Note* Table represents an descriptive analysis of extracted unobserved factors. Description column presents author's proposition of interpretation of those factors. Factor ID contains name of the Google Trend group from which given factors were extracted and identification number. Values in squared brackets next to query phrase represent the lag. Multiple numbers indicate multiple lags were selected

*Source* Own

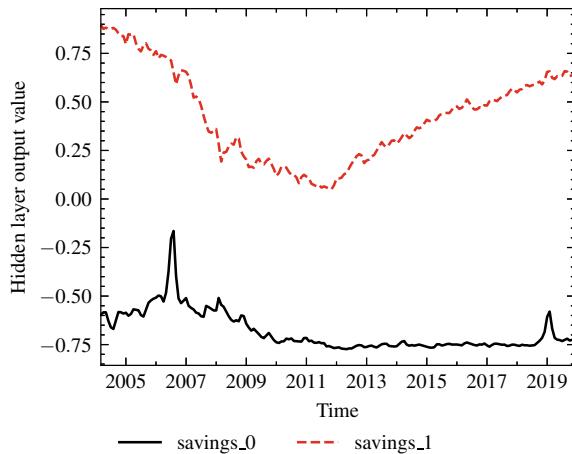
suggests that price perception is mainly driven by oil and gas. Price-related factors are plotted in Fig. 13.1.

$Savings_0$  represents perception on returns from financial instruments and funds, whereas  $savings_1$  describes credit cards perception. In both cases, some queries related to well-established banks. The author has found nothing outstanding in the results of savings factor drivers. Savings-related factors are plotted in Fig. 13.2.

Capital factors heavily focus on bear market anticipation.  $Capital_0$  describes the investment sentiment present on the market.  $Capital_1$  takes a possibility of bearish market occurrence, as the sole Google Trends query “bearish market”.  $Capital_2$  includes trends in the real estate market with respect to a potential economic downturn.  $Capital_3$  represents an interest in new companies’ entrance into the stock market.  $Capital_4$  combines stock market perception. Although the results are sensible in the author’s opinion, there is a risk of overestimating real estate’s importance in recession occurrence. As Google Trends is available from 2004, the only crises Google Trends has faced are the Great Recession and Solvency



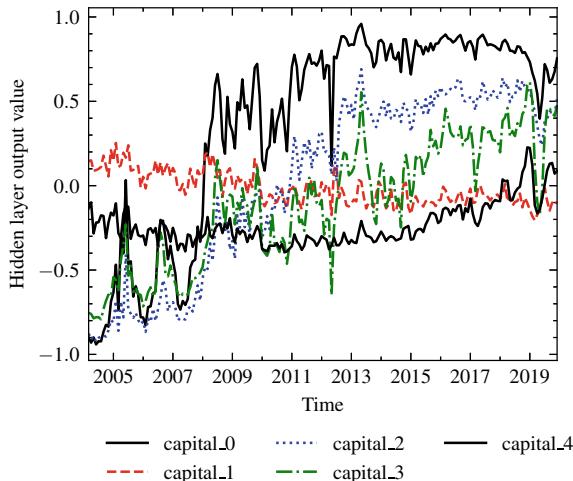
**Fig. 13.1** Hidden factors extracted from price-related Google Trends queries (Note Lines represent values of hidden layer aggregating the employment-related Google Trends queries. Solid-black lines represent sentiment towards oil market; red-dashed line represents sentiment towards gas market; blue-dotted line represents sentiment towards car market. *Source* Own) (Color figure online)



**Fig. 13.2** Hidden factors extracted from savings-related Google Trends queries (Note Lines represent values of hidden layer aggregating the savings-related Google Trends queries. Solid-black lines represent values of the factor related to financial instruments; red-dashed line represents the sentiment towards credit card. *Source* Own) (Color figure online)

Crisis, and recently, COVID-19 and Russo-Ukrainian war. As the two latter crises have not been fully resolved by writing this chapter, they are neither included in the validation nor test period. Hence, the model captures the recession occurrence on the specification of the Great Recession. That issue is discussed further in the article as it might affect the forecasting capabilities of the model. Capital-related factors are plotted in Fig. 13.3.

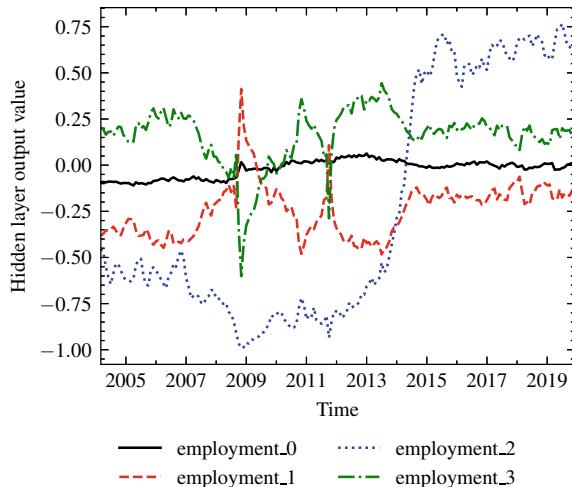
Insurance company searches heavily influence employment factors.  $employment_0$  captures the current unemployment situation,  $employment_1$  shows career seek and opportunities,  $employment_2$  represents the core, more difficult employment situation on the job market, and  $employment_3$  focuses solely on the well-established company. Hence, employment factors have layered a job market situation on its severity: whether the disturbances are temporal or generic. Employment-related factors are plotted in Fig. 13.4.



**Fig. 13.3** Hidden factors extracted from capital-related Google Trends queries (Note Lines represent values of hidden layer aggregating the capital-related Google Trends queries. Solid-black lines represent values of investment sentiment; red-dashed line represents the bearish market sentiment; blue-dotted line represents real estate market condition; green-dotdashed line represents IPO opportunities. *Source Own*) (Color figure online)

### *Importance on CPI Prediction*

The LSTVAR-ANN hybrid model allows us to evaluate which individual Google Trends queries and unobserved factors are the most influential ones. Importance is measured with the standard deviation of ceteris-paribus prediction. The author sets the input vector to be equal to an average value of each input variable. Then, one by one, author is changing the values of input  $x_i$  and records a new prediction of the CPI. The higher the standard deviation, the more impact an input variable has on the final outcome. Because each Google Trends query is normalised in value [0, 1], it is possible to uniformly draw N values and apply the same values for each query without losing any generalisation. By treating each Google Search query identically, the author eliminates an error which might come from different draws (Table 13.2).



**Fig. 13.4** Hidden factors extracted from employment-related Google Trends queries (*Note* Lines represent values of hidden layer aggregating the employment-related Google Trends queries. Solid-black lines represent values of current employment sentiment; red-dashed line represents career seek effort; blue-dotted line represents job market slowdown; green-dotdashed line represents interest in insurance companies. *Source* Own) (Color figure online)

They are visible leading Google Trends queries. Most prominent are unemployment-related phrases and the name of a well-established bank, followed by credit card and S &P500. However, the first five are taking the lion's share of the driving force, as the sixth one is 60% less influential than the leading one. The importance helps to prioritise Google Trends in case of dimension reduction. The author would like to point out that medium-level values do imply uselessness. Those queries might contain some information missing in leading queries, and ANN picks up those pieces of information. However, queries with low importance can be dropped due to a negligible impact on the final output.

It is possible to record inputs' importance over time. The LSTVAR-ANN model can be rerun in an expanding-window manner. For each draw, importance of factors can be established, and based on the ranking of the factor, one can trace its evolution over time. The author has expanded the window by every quarter. The starting window can include data within a time period of March 2004–December 2014. There were 20

**Table 13.2** Top 10 important Google Trends queries

<i>Query</i>	<i>Importance</i>
citibank [-2]	1.00
career [-1]	0.80
bailout [-1]	0.65
citibank [-1]	0.53
unemployment [-1]	0.40
career [-2]	0.38
citibank [0]	0.37
credit card [-2]	0.36
SP500 [0]	0.36
SP500 [-2]	0.35

*Note* Table shows importance values for the top 15 most important Google Trends queries. Values are normalised to the best-performing query (=1). Values in squared brackets represent the lag of a given query

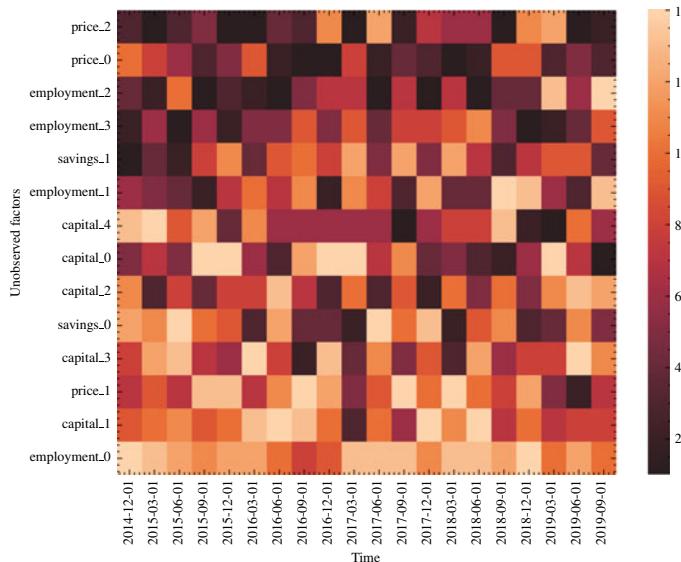
Ratio of importance metric between two variables shows the relative importance, e.g. citibank [-1] is almost twice less impactful than citibank [-2]

*Source* Own

expansions, with the last time period ending December 2019. The results are presented in Fig. 13.5. The dark cells represent high importance, whereas the lighter the cell, the less important it is.

Although some disturbances of factor position can be found, there are generally three groups of factors based on their importance—fundamental, contesting and low importance. The first four factors presented in Fig. 13.5 were usually at the top of the importance ranking. Those factors are the car market ( $prices_2$ ), oil market ( $prices_0$ ), job market slowdown ( $employment_2$ ) and insurance ( $employment_3$ ).

The least important factors were: current unemployment trends ( $employment_0$ ), bear market ( $capital_1$ ), gas market ( $prices_1$ ) and IPO occurrences ( $capital_3$ ). Low importance might come from slight variations in current unemployment trends and the bear market. IPO occurrences and, to a lesser extent, gas market are strongly correlated with other factors, which makes their marginal value added to the model small.



**Fig. 13.5** Ranks of factors for each expanding window draw (*Note* Matrix represents the importance rank of each unobserved factor. The darker the colour, the more important the factor was in a given period. On the y-axis, reader can read the factor ID—for full description the author refers to Table 13.1. *Source Own*)

### Transition Function Output

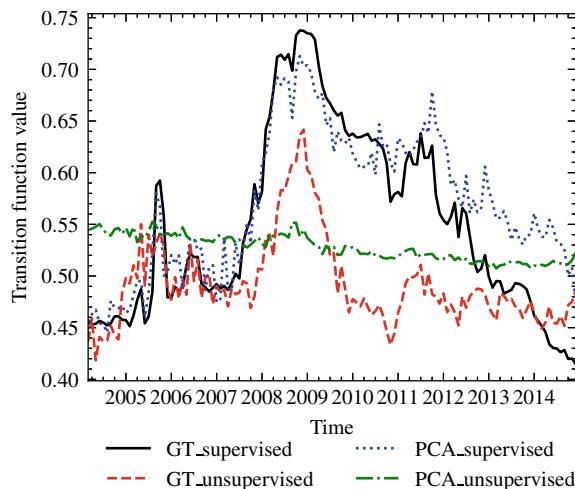
By extracting the value of the transition function, the author can track the regime occurrence. The author compares the results of 4 LSTVAR-ANN model types: supervised/unsupervised and Google Trends/PCA input. The transition function results are checked for two training periods: 2004–2014 and 2004–2016.

Both sentiment-supervised models provide similar results, which suggests that unobserved factors drive market sentiments. What is promising is that the results of the sentiment-unsupervised model with Google Trends input correlate with the results of supervised models (Pearson's correlation: 0.9). However, it gets flat after the Sovereign Crisis. Nonetheless, results suggest that Google Trends is a good indicator of market sentiments. The sentiment-unsupervised model with PCA

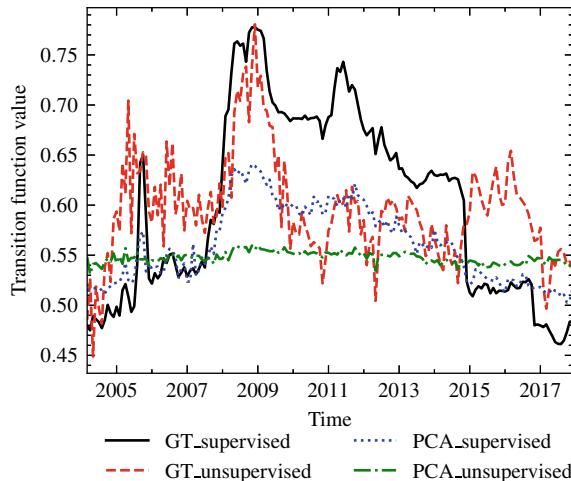
does not find any disturbances in regime ratio, which might suggest that PCA's information loss was too severe to model the transition variable. Results are presented in Fig. 13.6.

Similar results are found for the more extended training period. Both sentiment-supervised models have similar transition values. However, a model with PCA input shows minor volatility. The sentiment-unsupervised model with Google Trends input strongly correlates with the supervised counterpart (Pearson's correlation: 0.92). However, transition values show changes towards lower regimes in 2015, for which the author has yet to find an explanation. The sentiment-unsupervised model with PCA still does not find any disturbances in the regime ratio.

Results are presented in Fig. 13.7.



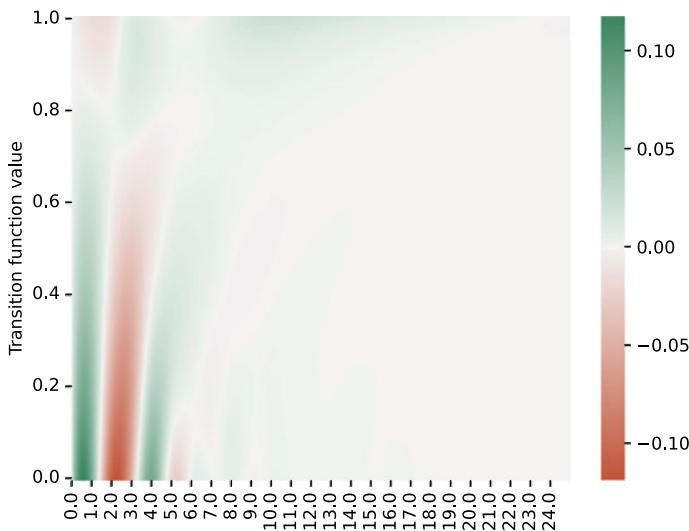
**Fig. 13.6** Transition values: period 2004–2014 (Note Lines represent the value of the transition function in each period for each LSTVAR-ANN model: Google Trends with supervised time series [solid black], Google Trends without supervised time series [dashed red], PCA with supervised time series [dotted blue], PCA without supervised time series [dotdashed green]. *Source* Own) (Color figure online)



**Fig. 13.7** Transition values: period 2004–2016 (Note Lines represent the value of the transition function in each period for each LSTVAR-ANN model: Google Trends with supervised time series [solid black], Google Trends without supervised time series [dashed red], PCA with supervised time series [dotted blue], PCA without supervised time series [dotdashed green]. *Source* Own) (Color figure online)

### *Response of CPI on Interest Rate Shock*

The 3D IRF graph represents the impulse response function values for each horizon (x-axis) for each transition value (y-axis). The green colour represents positive values, whereas red represents negative ones—the more intense the colour, the higher the absolute value. The results are consistent with the results obtained by LSTVAR estimated with grid search (Chojnowski, 2022). In Fig. 13.8 we observe a sudden increase for lower regimes followed by a substantial decrease in the first quarter. Then, the response smooths out. For higher transition values, the amplitude of spike and fall in Q1 diminishes, and for higher regimes, firstly, we observe the negative response followed by an extended positive reaction. The results suggest that the impact of interest rates on inflation varies depending on market sentiment values (here: Customer Confidence



**Fig. 13.8** 3D IRF of interest impulse on CPI (*Note* Graph represents a 3D-IRF of interest rate on CPI. X-axis represents number of month after the increase in interest rate occurs. Y-axis represents the value of the transition function, identifying the regime in which economy operates. Green colour indicates an increase of prices m/m, red colour a decrease. The more intense the colour, the higher the magnitude of change is recorded given month. *Source* Own) (Color figure online)

Index). Therefore, when predicting the impact of possible monetary policy changes, economists should include market sentiments in their forecasts.

### *Forecasting*

This section is divided into two parts—the first focuses on forecasting accuracy in pre-COVID-19 years (2017–2019), in which the economy was relatively stable; the second focuses on forecasting accuracy in the COVID-19 period (2020–2021) and dwells into the perturbances period.

Two horizons are checked—short term (1 month ahead) and long term (12 months ahead). The author assumes that in period  $t$ , Google Trends is known before realising macroeconomic fundamentals.

The long-term forecast is calculated recurrently—once 1 month ahead forecast is calculated, the predictions are added as an additional row in the input data, and the forecast is calculated again for another horizon (in this case, 2 months ahead).

Due to the inability to ensure regime stability, author is not using the LSTVAR coefficient matrix to calculate the longer-horizon forecasts.

For simplicity of the exercise, we assume Google Trends forecast to be naïve. Following the literature critique, the author has included five benchmarking models: naïve, the autoregressive process of order 1 AR(1), best-performing ARIMA based on validation period performance, exponential smoothing based on validation period performance and vector autoregressive model VAR( $p$ ), where  $p$  is chosen based on Schwarz Information Criteria (BIC) on combined training and validation period. The point forecast performance is measured with root mean squared error (RMSE), although, for legibility, results are normalised to the best-performing model.

### *Pre-COVID-19 Period*

For pre-COVID-19 forecasting performance, the author has chosen the training period 2004–2014, the validation period 2015–2016 and the test period 2017–2019.

For both horizons, ARIMA model was the best-performing one, barely beating naive and AR(1). Exponential smoothing was slightly worse performing than ARIMA in the short horizon. However, in the long run, it was off. Neural networks have performed worse than time series models. However, as the literature suggested, networks have performed relatively better in the long run. Not enough, though, to beat the ARIMA model (Table 13.3).

The complexity of the model might be to blame. With many more parameters to estimate compared to econometric models, a neural network might need more data to estimate them efficiently. Moreover, with fewer output variables, unsupervised counterparts of neural networks have lower accuracy. However, unsupervised models were more accurate for long-term horizons, moving them closer to econometric models.

**Table 13.3** RMSE of selected models: pre-COVID-19 scenario

Model	1-month forecast	12-month forecast
Naïve	1.02	1.01
AR(1)	1.02	1.00
Exp. Smooth.	1.09	1.54
ARIMA	1.00	1.00
VAR	7.71	4.37
<i>LSTVAR-ANN</i>		
Supervised with queries	2.05	1.45
Unsupervised with queries	2.83	1.17
Supervised with PCA	2.63	1.54
Unsupervised with PCA	3.40	1.29

*Note* Values are normalised to best-performing ARIMA model (=1). Values lower than 1 indicates better performance than ARIMA

*Source* Own

Additionally, neural networks with PCA inputs performed worse than neural networks aggregating Google Trends “by themselves”. Information loss in the PCA algorithm brings higher errors than the complexity of the aggregation part of the neural network.

#### *COVID-19 and Post-COVID-19 Period*

For pre-COVID-19 forecasting performance, the author has chosen the training period 2004–2016, validation period 2017–2019 and test period 2020–2022 (until September).

In the case of post-COVID-19 period forecasting, econometric models still outperform neural networks, with ARIMA and AR(1) being the most accurate ones. The neural networks needed to be more flexible to accommodate the sudden economic change caused by the lockdown. Interestingly, the unsupervised method with Google Trends input has outperformed the supervised counterpart in the short run. It suggests that neural networks have learnt the pattern before the COVID-19, but

**Table 13.4** RMSE of selected models COVID-19 scenario

Model	1-month forecast	12-month forecast
Naïve	1.28	1.02
AR(1)	1.04	1.00
Exp. Smooth.	1.39	1.27
ARIMA	1.00	1.01
VAR	3.93	2.06
<i>LSTVAR-ANN</i>		
Supervised with queries	4.09	1.83
Unsupervised with queries	3.14	1.86
Supervised with PCA	3.63	1.53
Unsupervised with PCA	3.82	1.91

*Note* Values are normalised to best-performing ARIMA model (=1). Values lower than 1 indicates better performance than ARIMA

*Source* Own

with changes in customer behaviour caused by lockdown changes, the priorities have also changed. Hence, the neural network has not adapted. As PCA has generalised the information contained in Google Trends by dimension reduction, the rules created by neural networks are also more generalised. In a stable time period (see Table 13.4), the loss of information disabled the model from creating more detailed relations between Google Trends and the economy. However, in an uncertain time, those generalised rules provided smaller errors than more detailed ones. The Great Depression and Solvency crisis periods were insufficient to explain the changes during COVID-19.

## CONCLUSION

In this chapter, the author has presented the features and analytic capabilities of the LSTVAR-ANN hybrid model. The author has presented extensive insights into market sentiments' nature and their impact on monetary policy. In the model, the author has assumed there are two regimes on the market defined by customer confidence. Thus, one regime describes the economy under optimism and forward-looking and the second one, under pessimism and past-conscious. The final output is the

linear combination of those two regimes. The model allows the ratio between those two regimes to change over time. The author assumes that market sentiments affect this ratio. Because market sentiments cannot be observed directly, the author proposed to use Google Trends queries to extract market sentiment values for the model. LSTVAR-ANN architecture allows the implementation of this model under one optimisation process.

Although the model results provide a better understanding of inflation behaviour and relationships, forecasting performance is still an area to develop. The author identifies three crucial points of such results. The first one concerns Ockham's razor. Econometric models are more straightforward than ANN. Therefore, the smaller dataset can estimate their parameters better. ANN has a much larger number of parameters; hence, the risk of overestimation is still valid, even though specific countermeasures have been implemented, such as dropout.

The second reason might be Google Trends' time period availability. Google Trends started to be reported in January 2004. Hence, not many Business Cycles have been recorded since then. Therefore, the model learns recession occurrences based only on the Great Recession and Solvency Crisis. This results in misinterpreting the causal effects between customer behaviour and crisis occurrence, leading to missing the upcoming recession (here: COVID-19 pandemic).

The last concern lies in Google Trends themselves. Google Trends shows only the number of queries of a given phrase. However, the analyst is blind to whether it was caused by negative or positive news. Hence, additional input or restrictions would be required for better mapping the Google Trends database to market sentiment values.

The proposed solution to the points mentioned above might be an extensive sentiment factor analysis from the LSTVAR-ANN model. A better understanding of those factors and their correlation with other economic data might provide a more accessible dataset with a more extended history than Google Trends. Having the economic dataset for market sentiment input can lower the number of parameters in the model and enlarge the dataset from which neural networks can learn. More recessive and extensive periods within the training dataset might bring better forecasting capabilities of the model.

Another part of development that the author would like to mention is the neural network architecture. The author proposed a sequential model with a small number of hidden values. Implementing the LSTM network in the Google Trends aggregation part might improve accuracy. Additionally, implementing a recurrent neural network in the economic part of the model can also bring benefits to the forecasts produced.

More research needs to be done on the network architecture of the LSTVAR-ANN model and its impact on forecasting. Another aspect worth mentioning is the lack of regime stability assumption in the LSTVAR-ANN model. LSTVAR model with grid search optimisation method can easily exclude models for which extreme regimes are unstable. Even though the presented model has stable solutions, in general, the LSTVAR-ANN model has not penalised unstable solutions.

Nonetheless, the author strongly believes in LSTVAR-ANN's powerful insights into forecasting and policy-making literature encourages readers to pursue their research in this area.

**Acknowledgements** This project was financed by the National Science Centre, Poland, grant No. 2017/25/N/HS4/02344.

## APPENDIX

See Fig. 13.9 and Table 13.5.



**Fig. 13.9** Architecture of sentiment-supervised LSTVAR-ANN hybrid model (*Note* Schema represents the architecture of the LSTVAR-ANN model used in this chapter. The top part represents the aggregation part of the Google Trends into unobserved factors. Middle part represents transforming unobserved factors into market sentiment. For “supervised” models, the layer “sentiment-out” is the output layer. The bottom part represents the LSTVAR model. *Source* Own)

**Table 13.5** List of used Google Trends queries

<i>Prices</i>	<i>Savings</i>	<i>Capital</i>	<i>Employment</i>
oil prices	bank	Nasdaq	unemployment
gas prices	interest rate	sp500	benefits
Texaco	savings	shares	jobs
SoCal	returns	IPO	pension
Target	chase	bullish market	bailout
Walmart	Wells Fargo	bearish market	bankruptcy
discount	credit card	bull	career
car prices	Citibank	bear	hire
house prices	401k	mortgage	insurance
prices	fund rate	investment	State Farm
inflation	mutual funds	real estate	Liberty Mutual
wages		estate agents	Berkshire Hathaway
VAT			
taxes			

*Source* Own

## REFERENCES

- Algaba, A., Borms, S., Boudt, K., & Van Pelt J. (2020). *The economic policy uncertainty index for Flanders, Wallonia and Belgium*. BFW digitaal/RBF numérique, 6.
- Arnerić, J., & Šestanović, T. (2021). Can recurrent neural networks predict inflation in euro zone as good as professional forecasters? *Mathematics*, 9(19), 1–13. <https://doi.org/10.3390/math9192486>
- Askitas, N., & Zimmermann, K. F. (2009). Google econometrics and unemployment forecasting. *Applied Economics Quarterly*, 55(2), 107–120.
- Baker, S. R., Bloom, N., & Davis, S. J. (2016). Measuring economic policy uncertainty. *The Quarterly Journal of Economics*, 131(4), 1593–1636. <https://doi.org/10.1093/qje/qjw024>
- Bantis, E., Clements, M. P., & Urquhart, A. (2022). Forecasting GDP growth rates in the United States and Brazil using Google Trends. *International Journal of Forecasting*. <https://doi.org/10.1016/j.ijforecast.2022.10.003>
- Barreira, N., Godinho, P., & Melo, P. (2013). Nowcasting unemployment rate and new car sales in south-western Europe with Google Trends. *NETNOMICS: Economic Research and Electronic Networking*, 14(3), 129–165.
- Bicchal, M., & Raja Sethu Durai, S. (2019). Rationality of inflation expectations: An interpretation of Google Trends data. *Macroeconomics and Finance in Emerging Market Economies*, 12(3), 229–239. <https://doi.org/10.1080/17520843.2019.1599980>
- Bleher, J., & Dimpfl, T. (2022). Knitting multi-annual high-frequency Google Trends to predict inflation and consumption. *Econometrics and Statistics*, 24, 1–26. <https://doi.org/10.1016/j.ecosta.2021.10.006>
- Bolboaca, M., & Fischer, S. (2019). *News shocks: Different effects in boom and recession?* (Technical Report 19.01). Swiss National Bank, Study Center Gerzensee.
- Cerqueira, V., Torgo, L., & Soares, C. (2022). Machine learning vs statistical methods for time series forecasting: Size matters. *Journal of Intelligent Information Systems*, 59(2), 415–433. <https://doi.org/10.1007/s10844-022-00713-9>
- Choi, H., & Varian, H. (2012). Predicting the present with Google Trends. *Economic Record*, 88, 2–9.
- Chojnowski, M. (2022). Monetary policy under continuous market sentiment regimes. In *Economic tendency surveys and economic policy-measuring output gaps and growth potentials*. Wydawnictwo Uniwersytetu Ekonomicznego w Poznaniu.
- Chojnowski, M., & Dybka, P. (2017). Is exchange rate moody? Forecasting exchange rate with Google Trends data. *Econometric Research in Finance*, 2(1), 1–21.

- Choudhary, M. A., & Haider, A. (2012). Neural network models for inflation forecasting: An appraisal. *Applied Economics*, 44(20), 2631–2635.
- Coble, D., & Pincheira, P. (2017). Nowcasting building permits with Google Trends. University Library of Munich. <https://EconPapers.repec.org/RePEc:pra:mprapa:76514>
- D'Amuri, F., & Marcucci, J. (2017). The predictive power of Google searches in forecasting US unemployment. *International Journal of Forecasting*, 33(4), 801–816.
- Dietzel, M. A. (2016). Sentiment-based predictions of housing market turning points with Google Trends. *International Journal of Housing Markets and Analysis*, 9(1), 108–136. <https://doi.org/10.5283/epub.33665>
- Drozdowicz-Bieć, M. (2011). Psychologiczne uwarunkowania nastrojów konsumentów i propozycja alternatywnego ich pomiaru. In *Zmiany aktywności gospodarczej w świetle wyników badań koniunktury*. Prace i Materiały IRG SGH.
- Gefang, D., & Strachan, R. (2009). Nonlinear impacts of international business cycles on the UK-A Bayesian smooth transition VAR approach. *Studies in Nonlinear Dynamics & Econometrics*, 14(1), 1–33.
- Giannone, D., Reichlin, L., & Small, D. (2008). Nowcasting: The real-time informational content of macroeconomic data. *Journal of Monetary Economics*, 55(4), 665–676.
- Gorodnichenko, Y., Tho, P., & Talavera, O. (2021). *The voice of monetary policy*. National Bureau of Economic Research. [https://www.nber.org/system/files/working\\_papers/w28592/w28592.pdf](https://www.nber.org/system/files/working_papers/w28592/w28592.pdf)
- Hasan, N. (2020). A methodological approach for predicting COVID-19 epidemic using EEMD-ANN hybrid model. *Internet of Things*, 11. <https://doi.org/10.1016/j.iot.2020.100228>
- Hinterlang, N. (2020). *Predicting monetary policy using artificial neural networks* (Technical Report 44). Deutsche Bundesbank Discussion Papers. <https://EconPapers.repec.org/RePEc:zbw:bubdps:4420201>
- Kropiński, P., & Anholcer, M. (2022). How Google Trends can improve market predictions—The case of the Warsaw Stock Exchange. *Economics and Business Review*, 8(1). <https://doi.org/10.18559/ebr.2022.2.2>
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS One*, 13(3). <https://doi.org/10.1371/journal.pone.0194889>
- McLaren, N., & Shanbhogue, R. (2011). Using internet search data as economic indicators. *Bank of England Quarterly Bulletin*, 51(2), 134–140.
- Moat, H. S., Curme, C., Avakian, A., Kenett, D. Y., Stanley, H. E., & Preis, T. (2013). Quantifying Wikipedia usage patterns before stock market moves. *Scientific Reports*, 3(1). <https://doi.org/10.1038/srep01801>

- Moshiri, S., & Cameron, N. (2000). Neural network versus econometric models in forecasting inflation. *Journal of Forecasting*, 19(3), 201–217.
- Naccarato, A., Falorsi, S., Loriga, S., & Pierini, A. (2018). Combining official and Google Trends data to forecast the Italian youth unemployment rate. *Technological Forecasting and Social Change*, 130, 114–122.
- Nagao, S., Takeda, F., & Tanaka, R. (2019). Nowcasting of the US unemployment rate using Google Trends. *Finance Research Letters*, 30, 103–109. <https://doi.org/10.1016/j.frl.2019.04.005>
- Ogbonna, A. E., Salisu, A. A., & Adewuyi A. (2020). Google Trends and the predictability of precious metals. *Resources Policy*, 65. <https://doi.org/10.1016/j.resourpol.2019.101542>
- Panda, R. K., Pramanik, N., & Singh, A. (2011). Daily river flow forecasting using wavelet ANN hybrid models. *Journal of Hydroinformatics*, 13(1), 49–63.
- Peersman, G., & Smets, F. (2001). *The monetary transmission mechanism in the euro area: More evidence from VAR analysis* (Working paper, Series). European Central Bank.
- Peirano, R., Kristjanpoller, W., & Minutolo, M. C. (2021). Forecasting inflation in Latin American countries using a SARIMA-LSTM combination. *Soft Computing*, 25(16), 10851–10862.
- Petropoulos, A., Siakoulis, V., Stavroulakis, E., Lazaris, P., & Vlachogiannakis, N. (2022). Employing Google Trends and deep learning in forecasting financial market turbulence. *Journal of Behavioral Finance*, 23(3), 353–365. <https://doi.org/10.1080/15427560.2021.1913160>
- Preis, T., Moat, H. S., & Stanley, H. E. (2013). Quantifying trading behavior in financial markets using Google Trends. *Scientific Reports*, 3(1), 1–6.
- Qiao-Feng, T., Xiao-Hui, L., Xu, W., Hao, W., Xin, W., Yi, J., & Ai-Qin, K. (2018). An adaptive middle and long-term runoff forecast model using EEMD-ANN hybrid approach. *Journal of Hydrology*, 567, 767–780.
- Shafaei, M., Adamowski, J., Fakheri-Fard, A., Dinpashoh, Y., & Adamowski, K. (2016). A wavelet-SARIMA-ANN hybrid model for precipitation forecasting. *Journal of Water and Land Development*, 28, 27.
- Singhania, R., & Kundu, S. (2020). Forecasting the United States unemployment rate by using recurrent neural networks with Google Trends data. *International Journal of Trade, Economics and Finance*, 11(6), 135–140.
- Sorić, P., & Lolić, I. (2017). Economic uncertainty and its impact on the Croatian economy. *Public Sector Economics*, 41(4), 443–477.
- Stempel, D., & Zahner, J. (2022). *DSGE models and machine learning: An application to monetary policy in the Euro area*. MAGKS Papers on Economics. <https://ideas.repec.org/p/mar/magkse/202232.html>

- Szafranek, K. (2019). Bagged neural networks for forecasting Polish (low) inflation. *International Journal of Forecasting*, 35(3), 1042–1059. <https://doi.org/10.1016/j.ijforecast.2019.04.007>
- Teräsvirta, T., Yang, Y., et al. (2014). *Specification, estimation and evaluation of vector smooth transition autoregressive models with applications* (CREATES Research Papers 2014-08).
- Tkacz, G. (2001). Neural network forecasting of Canadian GDP growth. *International Journal of Forecasting*, 17, 57–69.
- Wang, L., Zou, H., Su, J., Li, L., & Chaudhry, S. (2013). An ARIMA-ANN hybrid model for time series forecasting. *Systems Research and Behavioral Science*, 30(3), 244–259. <https://doi.org/10.1002/sres.2179>
- Weise, C. L. (1999). The asymmetric effects of monetary policy: A nonlinear vector autoregression approach. *Journal of Money, Credit and Banking*, 31, 85–108.
- Yu, L., Zhao, Y., Tang, L., & Yang, Z. (2019). Online big data-driven oil consumption forecasting with Google trends. *International Journal of Forecasting*, 35(1), 213–223.
- Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, 159–175.



## CHAPTER 14

---

# The FVA Framework for Evaluating Forecasting Performance

*Michael Gilliland*

The last decade has been an exciting and fruitful time for the advancement of forecasting. Traditional time series methods have been enhanced, and in some cases supplanted, by a new generation of data scientists bringing new approaches from machine learning and artificial intelligence. But this rapid innovation has fueled claims of performance improvement that require proper assessment. The forecast value added (FVA) framework provides an alternative to traditional methods for assessing forecasting performance.

---

M. Gilliland (✉)  
International Institute of Forecasters, Seagrove, NC, USA  
e-mail: [mike.gilliland@forecasters.org](mailto:mike.gilliland@forecasters.org)

## WHAT IS FVA?

Forecast value added (FVA) provides a framework for identifying waste, inefficiency, and bad practices in a forecasting process. It is not a measure of forecast accuracy, but rather a reflection of the efficacy of the overall process. By definition (Gilliland, 2002):

FVA ≡ The change in a forecasting performance measure that can be attributed to a particular step or participant in the forecasting process

FVA assesses whether our forecasting efforts are improving accuracy, are having no discernable effect on accuracy, or are just making it worse. Note that while FVA can be used with any forecasting performance measure, including bias, this chapter will only be discussing FVA with accuracy measures.

Consider a very simple forecasting process that has just two steps:

1. Modeling (to create the “computer-generated forecast”)
2. Management review (to create the “judgmentally adjusted forecast”).

Suppose that the computer-generated forecast achieved accuracy of 70%, while the judgmental adjustment raised accuracy to 75%. We would then say that the management review step had positive “value added” of five percentage points. When a process step makes the forecast less accurate, the value added is negative.

While FVA is most commonly used to assess the impact (positive or negative) of judgmental overrides to a computer-generated forecast, it is equally well suited to evaluate the impact from modeling efforts. It does this by comparing performance of the computer-generated forecast to a naïve forecast or to other alternative models. FVA seeks to isolate the impact of sequential activities in a forecasting process so that the process can be streamlined and tuned, making it more efficient and more effective.

While FVA is sometimes misunderstood to be a financial measure, it is not. FVA evaluates the change in accuracy of a point forecast, saying nothing of the business value or monetary impact derived from the change. Whether accuracy improvement leads to business value is an important, yet entirely separate question. This is addressed by Robette

(2023) and other contributors to a special feature section on “Does forecast accuracy even matter?” in Issue 68 of *Foresight: The International Journal of Applied Forecasting*.

## WHY USE FVA?

Traditional performance measures such as mean absolute error (MAE), mean absolute percentage error (MAPE), and its various flavors like weighted MAPE (wMAPE) and symmetric MAPE (sMAPE) indicate the accuracy of a point forecast, but little else. Knowing that MAPE is 30% tells us the magnitude of our error but provides no informative context. There is no indication of whether 30% MAPE is good or bad, or how this compares to what alternative methods would achieve. Furthermore, traditional measures do not indicate *forecastability*—the level of accuracy we *should* be able to reach given the nature of what we are forecasting. (While estimating forecastability is not addressed in this chapter, it can be approached through the measure of entropy; Catt, 2014). Also, traditional measures tell us nothing about the overall efficiency of our forecasting process, i.e., whether process steps add value by improving accuracy, or are non-value adding waste.

As part of a broad class of “relative” performance measures, the FVA framework steps in to address some of these gaps in traditional measures. Theil’s-U (Theil, 1966) was an early attempt to provide context to forecast performance by indicating accuracy relative to the out-of-sample, one-step-ahead naïve forecast. Mean absolute scaled error (MASE; Hyndman & Kohler, 2006) is a more recent example in this class of measures, indicating performance relative to the in-sample, one-step-ahead naïve forecast. What distinguishes FVA from other relative performance measures, however, is its focus on the *entire* forecasting process—not just the resulting forecast. FVA directs attention to each sequential step in a forecasting process to identify activities that may be ineffective (and thereby wasting resources) or are even degrading performance (and making the forecast less accurate). The FVA mindset is motivated by this objective:

To generate forecasts as accurate and unbiased as can reasonably be expected (given the nature of what is being forecast), and to do this as efficiently as possible.

## APPLYING THE SCIENTIFIC METHOD TO FORECASTING

FVA analysis is the application of the basic scientific method to the evaluation of forecasts. It can be understood by analogy to the way a new drug treatment would be evaluated.

Suppose a new pill has been developed to cure the common cold. It is claimed that 80% of patients will be symptom free within two weeks of taking it. Is this pill effective? The 80% cure rate sounds good until we realize that virtually all cold sufferers will be symptom free within two weeks anyway – without taking anything.

In pharmaceutical evaluation, the new treatment is compared to a placebo. Unless those taking the treatment have a better outcome than those not taking the treatment, we cannot conclude the treatment is effective.

In FVA analysis, our placebo is the naïve forecast. The steps in a forecasting process are essentially “treatments” applied to the forecast created in the prior step. There could be several such treatments in a multi-step process allowing review and adjustments by several layers of management. However, the simplest example is a judgmental override (such as adding 100 units) to a computer-generated forecast.

One popular way to show results of an FVA analysis for a sequential process is through a stairstep report. Table 14.1 shows a sample report using actual data reported by a large consumer goods manufacturer across their full product line (Schubert & Rickard, 2011).

At this organization, a naïve forecast achieved 60% accuracy and their forecasting system achieved 65% accuracy, resulting in +5 percentage points of “value added” by their modeling efforts. However, after

**Table 14.1** Stairstep report with FVA results

<i>Process step</i>	<i>Forecast accuracy (%)</i>	<i>FVA vs. Placebo (%)</i>	<i>FVA vs. computer (%)</i>
Placebo (Naïve Forecast)	60	–	–
Computer-Generated Forecast	65	5	–
Judgmental Override	62	2	–3

management reviewed the forecasts and made judgmental overrides, accuracy dropped to 62%. It is not unusual to find management adjustments resulting in negative value added, as in this case. Forecasting can be a highly politicized process within organizations. Those reviewing and making adjustments may have little training or skill in forecasting, or be motivated by aspirations or objectives other than accuracy.

In the Table 14.1 example, the initial computer generation of the forecast was treated as a single step, with its value added determined relative to the naïve forecast. But forecast modeling needs not always be treated as a single step. Petropoulos and colleagues (2018) showed how judgmental model selection can improve upon the automatic model selection of forecasting software (with an equally weighted combination of the judgmental model's and the automatic model's forecasts performing better still).

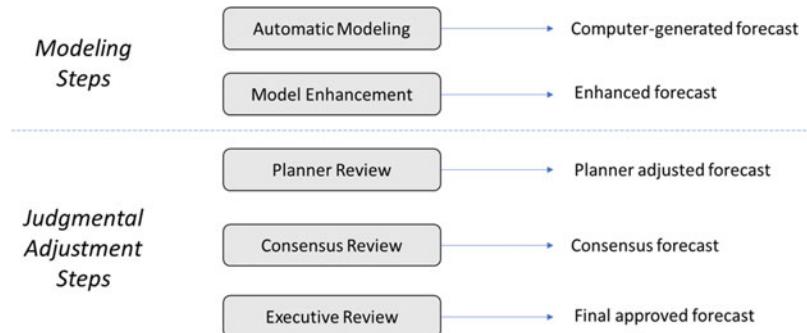
Also, a data scientist may build a model in steps, such as enhancing a basic time series model with exogenous variables. In this case, these are sequential steps that could be evaluated in the stairstep report. But in a machine learning setting, the stairstep format may not be as applicable. In machine learning, various modeling steps like feature selection and hyperparameter selection may be done simultaneously rather than sequentially, so could not be distinguished in the stairstep format.

## FVA ANALYSIS: STEP-BY-STEP

Having seen how FVA complements traditional forecasting performance measures to provide insights into a forecasting process, this section will focus on how to assemble the necessary data and conduct an FVA analysis, step by step.

### *Mapping the Process*

The first step in FVA analysis is to understand and map the overall forecasting process. The process may be simple, as with the two-step process shown in Table 14.1, with just a computer-generated forecast and a judgmental override. However, as shown in Fig. 14.1, the process can be much more elaborate, with considerable effort put toward modeling, as well as a consensus or collaborative step with participation from multiple internal departments, or even external participants like customers and suppliers. For instance, some organizations include a final executive review



**Fig. 14.1** Example of a multi-step forecasting process

and approval step. All these activities consume high-cost management time, so a fiscally responsible organization would use FVA to make sure these efforts are improving forecast accuracy as expected.

### *Collecting the Data*

The process map identifies steps in the sequential flow of forecast reviews and updates. The data for FVA analysis is the forecast created at each step, at the most granular level of detail (such as item at a location), for each time bucket (e.g., day, week, or month) in the available history. Forecasts should be recorded as of the standard lead time the organization uses to track forecasting performance. This is typically not the one-step-ahead forecast, since most organizations must make decisions days, weeks, or even months ahead of the period being forecast.

In addition to all the forecasts being recorded, FVA requires the “actual” values at the most granular level of detail for each time bucket. The actuals are needed for comparison to the forecasts.

### *Analyzing the Process*

Because FVA is concerned with the change in a performance measure, the analysis can proceed using whatever traditional performance measure is preferred. However, there are several considerations for the choice of measure, which will be discussed more thoroughly below. MAPE is the most popular choice—despite its many known weaknesses and calls for

its retirement (Tichy, 2023)—because it is easy to interpret and therefore commonly used by organizations for normal performance reporting. However, there are other more suitable options.

As shown in the stairstep report of Table 14.1, the choice of “placebo” plays a key role in FVA analysis. Normally, a basic naïve forecast, also referred to as random walk or “no-change” model, is used as the placebo and lagged according to the lead time (thus, if lead time is three weeks, the actual value from three weeks prior would be used as the forecast). When the data is seasonal, using a seasonal naïve as the placebo provides a more appropriate benchmark for evaluating performance.

Using a placebo is important because it delivers the level of accuracy an organization can achieve with little (or no) investment in forecasting. For example, the “no-change” forecast can be implemented without special software or the need for a skilled data scientist. Other simple, low-cost models that are suitable as placebos include a moving average or single exponential smoothing. A further option is a simple combination benchmark (such as an equally weighted average of single, damped trend, and Holt’s exponential smoothing) used in recent M competitions (Makridakis & colleagues, 2018). The M competitions, and empirical studies such as Morlidge (2014), show that even simple models can be surprisingly difficult to beat. For example, in Morlidge’s study of eight companies in consumer and industrial markets, 52% of their forecasts failed to exceed the accuracy of the no-change forecast.

### *Reporting the Results*

The two major questions addressed by FVA analysis are whether the overall process is performing better than the placebo (a naïve forecast or other simple benchmark), and more specifically, whether judgmental adjustments are adding value by improving accuracy compared to the computer-generated forecast. A negative answer to either question should prompt a review and overhaul of the forecasting process.

A more thorough FVA analysis will consider pair-wise comparisons of all the sequential steps in the process, considering such questions as:

- Is the default computer-generated forecast more accurate than the naïve?
- Do enhancements to the default model (such as incorporating exogenous variables) or alternative models improve upon the default model?
- Do planner adjustments to the forecast improve accuracy?
- Does the consensus process improve accuracy?
- Does the final executive review and adjustment improve accuracy?

It is likely to find that some of these steps add value, and others do not. It may be possible to improve the performance of some steps, such as through training of process participants or use of additional data in the models. Or we may decide to *remove* some steps from the process, because they are harming accuracy or they are not making a discernable difference. Note that steps with no discernable effect still have a cost in terms of time and resources spent and should therefore be considered for removal. By eliminating steps with no impact on accuracy, FVA delivers the benefit of a lower cost, more efficient process that consumes fewer resources—even if the process is not improving accuracy.

Although the stairstep format is an often-used example, there is no prescribed way to report FVA results. In whatever format, negative FVA findings are common and can be embarrassing to those process participants who are failing to add value. Since the objective of FVA analysis is to improve the forecasting process—not necessarily to embarrass anyone—results should be presented tactfully.

### *Interpreting the Results*

The FVA approach is intended to be objective and scientific, so it is important to avoid drawing conclusions that are not justified by the data. Over short time frames and limited data, a difference in accuracy may simply be due to chance.

As mentioned above, there are considerations in choosing the measure used in FVA analysis. Dayydenko and Fildes (2015) systematized the well-known problems with existing forecasting performance measures

and identified additional limitations. Their key finding was that different measures can lead to different conclusions being drawn from the same data. Thus, it may be advisable to replicate a FVA analysis with different measures to check the robustness of the results.

Goodwin (2018) noted that using relative error measures (like MASE) in an FVA analysis can lead to difficulties in interpretation, as these are already providing a comparison to naïve forecasts. However, citing Davydenko and Fildes, he recommends the average relative MAE for use when measuring FVA across different products (rather than for a single product). AveRelMAE is the geometric mean of relative MAE values, but unfortunately, is not generally available in commercial forecasting software.

Caution should be taken in interpreting results and particularly in making process changes based on the results. Even when there is value added by a process step, if the accuracy improvement is small, it may not be worth the cost of executing the step. Changes in accuracy are ultimately meaningless if they have no impact on decisions and actions.

## FVA CHALLENGES AND EXTENSIONS

### *Compete or Combine?*

FVA is characterized by the evaluation of sequential steps in a forecasting process. This puts each step at risk of being eliminated unless it improves upon its preceding step. But instead of viewing process steps as in competition with each other, Singh and colleagues (2015) suggest that combining the forecasts from each step can lead to a better result. Table 14.2 shows how this could happen.

Here, the computer-generated forecast has an absolute error of 7 units, and the adjusted forecasts by Sales, Marketing, and Executive reviews

**Table 14.2**  
Combination of  
non-value adding steps

	<i>Process step</i>	<i>Forecast</i>	<i>Error</i>	
	Computer	103	7	
	Sales Review	122	12	
	Marketing Review	97	13	Actual = 110
	Executive Review	126	16	
	Average of all forecasts	112	2	

are each less accurate (with absolute errors of 12, 13, and 16, respectively). FVA analysis would say these are non-value adding steps and can be eliminated from the process, leaving only the computer forecast. However, averaging the three review step forecasts with the computer forecast has an absolute error of 2, which is a considerable improvement over the computer-generated forecast alone. Although when a participant has advance knowledge that their forecast will be combined—rather than evaluated on its own—this could lead to gaming behavior and a lot of double guessing. This example shows that combinations may alter the way FVA analysis should be applied.

We know from the M competitions and other studies that averaging the forecasts from different methods will often result in improved accuracy. Atiya (2020) provides a recap of our knowledge about combining forecasts, deducing two facts that are also verified in empirical research and in most time series forecasting competitions:

1. Forecast combination should be a winning strategy if the constituent forecasts are either diverse or comparable in performance.
2. One should exclude forecasts that are considerably worse than the best ones in the pool unless they are very diverse from the rest.

Table 14.2 example, along with the recognized benefits of combining, adds another point of caution in interpreting FVA results. Even a process step that, by itself, is reducing accuracy, it may improve accuracy as part of a combination. Therefore, in such settings, stairstep evaluation may become misleading and require a more holistic assessment instead. As explained earlier, this can also be the case when using machine learning models; although a data pre-processing or feature-selection step may individually deteriorate performance, they could ultimately lead to accuracy gains once combined.

### *Stochastic Value Added (SVA): FVA for Probabilistic Forecasts*

The FVA framework was envisioned for use with point forecasts. But as quantile and other probabilistic forecasts achieve wider adoption, an extension of the framework has been proposed by de Kok (2017).

de Kok begins by defining a new total percentage error (TPE) measure to assess the difference between the actual and forecasted *distributions* of

demand. While more data hungry and complex-to-compute than other error measures, TPE expresses more fully the range of uncertainty in forecasts. This presumably enables better decision making, such as for inventory planning. Rather than planning for the middle-of-the-road point forecast, an organization can consider the full demand distribution to protect itself against specific risks or position itself to pursue fortuitous opportunities.

TPE also provides a more comprehensive way to gauge the quality of the forecast through the stochastic value added (SVA) measure. SVA measures improvement in the difference between actual and forecasted *distributions*. By evaluating the full distribution rather than just the point forecast, SVA provides a more complete picture of the value of a forecast.

## SUMMARY

The FVA framework is not so much a rigid set of rules and procedures as it is a mindset—a scientific mindset applied to the evaluation of forecasting performance. FVA draws attention away from the accuracy achieved by a forecasting process as accuracy is ultimately limited by the nature of what is being forecast. Instead, FVA refocuses that attention on the effectiveness of the process (how much it is improving accuracy versus alternatives like a naïve forecast) and its efficiency (how many resources are utilized in delivering that improvement).

FVA will expose fundamentally flawed processes that either fail to beat a naïve forecast, or involve judgmental adjustments that worsen the computer-generated forecast. These are the kind of findings that alert organizations to underlying bad practices. FVA complements traditional performance measures by identifying the waste and inefficiency that can plague any forecasting process.

## REFERENCES

- Atiya, A. F. (2020). Why Does Forecast Combination Work So Well? *International Journal of Forecasting*, 36(1), 197–200.
- Catt, P. (2014). Entropy as an A Priori Indicator of Forecastability. [https://www.researchgate.net/publication/268804262\\_Entropy\\_as\\_an\\_A\\_Priori\\_Indicator\\_of\\_Forecastability](https://www.researchgate.net/publication/268804262_Entropy_as_an_A_Priori_Indicator_of_Forecastability)
- Davydenko, A., & Fildes, R. (2015). Forecast Error Measures: Critical Review and Practical Recommendations, in Gilliland, M., Tashman, L, & Sgavlo, U.

- (eds.), *Business Forecasting: Practical Problems and Solutions*. Hoboken: John Wiley & Sons.
- de Kok, S. (2017). The Quest for a Better Forecast Error Metric: Measuring More Than Average Error, *Foresight: The International Journal of Applied Forecasting*, Issue 46, 36–45.
- Gilliland, M. (2002). Is Forecasting a Waste of Time? *Supply Chain Management Review*, 6(4), 16–23.
- Goodwin, P. (2018). *Profit From Your Forecasting Software*. John Wiley & Sons.
- Hyndman, R. J., & Koehler, A. B. (2006). Another Look at Measures of Forecast Accuracy. *International Journal of Forecasting*, 22(4), 679–688.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 Time Series and 61 Forecasting Methods. *International Journal of Forecasting*, 36(1), 54–74.
- Morlidge, S. (2014). Forecast Quality in the Supply Chain. *Foresight: The International Journal of Applied Forecasting*, Issue 33, 26–31.
- Petropoulos, F., Kourentzes, N., & Nikolopoulos, K. (2018). Judgmental Selection of Forecasting Models. *Journal of Operations Management*, 60, 34–46.
- Robette, J. (2023). Does Improved Forecast Accuracy Translate to Business Value? *Foresight: The International Journal of Applied Forecasting*, Issue 68, 12–19.
- Schubert, S. & Rickard, R. (2011). Using Forecast Value Added Analysis for Data-driven Forecasting Improvement, IBF Best Practices Conference.
- Singh, H., Raman, S. & Wilson, E. (2015). Practical Considerations in Forecast Value Added Analysis, *Foresight: The International Journal of Applied Forecasting*, Issue 38, 25–30.
- Theil, H. (1966). *Applied Economic Forecasting*. North-Holland.
- Tichy, M. (2023). Time to Retire the MAPE. *Foresight: The International Journal of Applied Forecasting*, Issue 69, 5–12.

# AUTHOR INDEX

## A

Abbasi, B., 336  
Abbass, H.A., 247  
Abbott, S., 156  
Abolghasemi, M., 314, 316, 322,  
    324, 326, 328  
Abreu, M., 229  
Ackermann, K., 117  
Adamowski, J., 342  
Adamowski, K., 342  
Adewuyi, A., 342  
Agarap A.F., 63  
Ahmed, H., 193  
Ahmed, R.A., 50, 56  
Aiolfi, M., 192  
Aiolli, F., 223  
Ai-Qin, K., 342  
Alaa, A., 297  
Alayon, D., 4  
Albanie, S., 238  
Alexandrov, A., 62  
Algaba, A., 342  
Alolayan, O.S., 68  
Amodei, D., 33

Amos, B., 321  
Anandarajan, M., 224  
Andrychowicz, M., 223  
Anholcer, M., 342  
Antiga, L., 291  
Antoniou, A., 223  
Anumanchipalli, G.K., 23  
Apiletti D., 49  
Arik, S.O., 114  
Arinze, B., 224  
Armstrong, J.S., 83, 224  
Arnerić, J., 341  
Arpin, D., 298  
Askitas, N., 342, 343  
Assimakopoulos, V., 52, 124  
Astashonok, Y., 298  
Atas, M., 330  
Athanasopoulos, G., 52  
Atiya, A.F., 265, 266, 382  
Aubet, F.X., 88  
Avakian, A., 342

## B

Babai, M.Z., 49

- Bader, D., 116  
 Bagnall, A., 256  
 Bahdanau, D., 289  
 Baier, L., 170  
 Bailey, J., 320  
 Bai, S., 62  
 Ba, J., 63  
 Balcan, M.F., 307  
 Baldo, F., 328  
 Balioglu, C., 298  
 Bandara, K., 68, 110, 115, 117  
 Bantis, E., 342, 344, 346  
 Barandas, M., 229  
 Barddal, J.P., 165  
 Barker, J., 50, 57, 193, 202  
 Barrat, J., 43  
 Barreira, M., 342  
 Barrow, D., 49, 192  
 Barrow, D.K., 192  
 Bartakke, P.P., 197  
 Bass, F.M., 85, 86  
 Bates, J.M., 51, 192  
 Bean, R., 324  
 Beaumont, A.N., 52  
 Bédubourg, G., 256  
 Belhadjali, M., 54  
 Bello, H., 334  
 Benitez, J.M., 52  
 Benedikter, R., 23  
 Bengio, S., 223  
 Bengio Y., 223  
 Bengio, Y., 64  
 Ben Taieb, S., 49  
 Bergmann, U., 304  
 Bergmeir, C., 52  
 Bertsimas, D., 321, 322, 326–328  
 Bessa, R.J., 49  
 Beygelzimer, A., 121, 309  
 Bica, I., 297  
 Bicchal, M., 342  
 Bijak, J., 49  
 Birr, S., 88  
 Bishop, C.M., 98  
 Bleher, J., 342  
 Bloom, N., 342  
 Bohlke-Schneider, M., 62  
 Bohte, S., 62  
 Bojer, C.S., 59  
 Bolboaca, M., 343  
 Bonissone, P.P., 223  
 Bontempi, G., 120  
 Borchert, O., 301, 302  
 Borms, S., 342  
 Borovsky, A., 62  
 Bostrom, A., 256  
 Bostrom, N., 42, 43  
 Bota, P., 229  
 Bouchon-Meunier, G.W., 247  
 Boudette, N., 16  
 Boudt, K., 342  
 Box, G.E.P., 108  
 Boylan, J.E., 49  
 Bradbury, J., 291  
 Brandom, R., 8, 12  
 Braun, N., 230  
 Brockman, J., 15  
 Brodwin, E., 21  
 Browell, J., 49  
 Brown, M., 8  
 Brown, T., 301  
 Brys, T., 335  
 Bucarey, V., 320  
 Budescu, D.V., 265  
 Budi, I., 225  
 Bumshik, L., 194  
 Burda, M., 112
- C**  
 Calantone, R.J., 88  
 Callot, L., 55  
 Cameron, N., 342  
 Campbell, S., 117  
 Cang, S., 266

- Canseco, M.M., 168  
 Cao, H., 165, 166  
 Cao, W., 52  
 Cappart, Q., 334  
 Carey, N., 5, 17  
 Carnevale, C., 49  
 Carpov, D., 61  
 Catt, P., 375  
 Cerqueira, V., 178, 225, 341  
 Chakravarty, S., 298  
 Challu, C., 168  
 Chanan, G., 291  
 Chang E.F., 23  
 Chang, M.W., 303  
 Chang, W.C., 62  
 Chan, J., 320  
 Chapados, N., 61  
 Chapman, B., 42  
 Chapman, L., 15  
 Chartier, J., 23  
 Chatigny, P., 57  
 Chatzos, M., 324, 328, 329  
 Chen, E., 265  
 Chen, K., 288  
 Chen, T., 253  
 Chen, W., 176  
 Chess, B., 301  
 Chetelat, D., 334  
 Child, R., 301  
 Choi, H., 343  
 Choi, J.Y., 194  
 Chojnowski, M., 343, 344, 346, 360  
 Cho, K., 289  
 Choudhary, M.A., 341, 342  
 Christensen, G., 41  
 Christiano, P., 33  
 Christ, M., 260  
 Chu, F., 169  
 Cirillo, P., 49  
 Claeskens, G., 52  
 Clarke, L., 21  
 Clements, M.P., 49  
 Cloutier, J., 64  
 Coble, D., 342  
 Collopy, F., 224, 252  
 Colmenarejo, S.G., 223  
 Cordeiro, C., 49  
 Coren, M.J., 13  
 Corrado, G., 288  
 Coskun, B., 298  
 Coyle, L., 166  
 Crone, S.F., 56, 225  
 Cunningham, P., 166  
 Curme, C., 342  
 Cyrino Oliveira F.L., 49
- D**
- Dafoe, A., 43  
 Dai, A.M., 223  
 Dai, H., 354  
 Dai, Q., 168  
 d'Alché-Buc, F., 121, 309  
 D'Amuri, F., 342  
 Dasgupta, S., 304  
 Da Silva, L.F., 102  
 Das, P., 298  
 Davis, E., 6, 15  
 Davis, S.J., 342  
 Davydenko, A., 380, 381  
 Dawid, A.P., 178  
 Dean, J., 288  
 De Baets, S., 49  
 De Barros, R.S.S., 169  
 de Bézenac, E., 304  
 De Gooijer J.G., 50, 53  
 De Kok, S., 382  
 Delany, S.J., 166  
 Delgado, J., 298  
 Demirović, E., 320  
 Dengel, A., 223  
 Denil, M., 223  
 De Raedt, L., 335  
 Detassis, F., 330

- Deuschel, J., 156  
 Devlin, J., 303  
 Dhariwal, P., 301  
 Dhillon, I.S., 114  
 Dieleman, S., 334  
 Dietze, M.A., 342  
 Dilkina, B., 334  
 Dimpfl, T., 342  
 Dinpashoh, Y., 342  
 Dipak, C.J., 86  
 Dokumentov, A., 49  
 Donti, P., 321, 329, 330  
 Doshi-Velez, F., 33  
 Drozdowicz-Bieć, M., 344  
 Dumonal, M., 116  
 Duncan, G.T., 112, 229  
 Dunn, J., 326, 328  
 Durai, S., 342  
 Dybka, P., 344
- Fernández, A., 179  
 Feuerriegel, S., 306  
 Fildes, R., 225, 226, 228–231, 238,  
     240, 245, 280, 282, 380, 381  
 Finley, T., 166, 168, 176, 237, 245,  
     300
- Fiordaliso, A., 103  
 Fioretto, F., 328, 329  
 Fischer, S., 343  
 Flunkert, V., 62, 116, 300, 303  
 Folgado, D., 230  
 Ford, M., 13, 15  
 Fortunato, M., 331  
 Fox, E., 114  
 Frauen, D., 306  
 Freitas, M., 225  
 Freitas, N.D., 223  
 Frejinger, E., 322  
 Friedman, J., 99, 182  
 Fulcher, B.D., 260

**E**

- Eberhard, A., 335  
 Eddy Patuwo, B., 54  
 Eisenach, C., 301, 303  
 Elmachtoub, A., 317, 318  
 Elyan, E., 333  
 Enembreck, F., 165, 170  
 Erkkilä, T., 54, 55, 86  
 Esmaeilbeigi, R., 316, 322, 324  
 Etherington, D., 21  
 Ettinger, S., 41  
 Evans, O., 43

**F**

- Fakheri-Fard, A., 342  
 Falorsi, S., 342  
 Fathi, K., 23  
 Felfernig, A., 330  
 Feng, L., xxii  
 Fergus, R., 63  
 Fernandes, L., 230

**G**

- Gaber, M.M., 168  
 Gabrys, B., 223, 225  
 Gallagher, J., 23  
 Gama, J., 169  
 Gamakumara, P., 68  
 Gamboa, H., 230  
 García, S., 179, 182  
 Garnett, R., 113, 114, 166  
 Garrabrant, S., 33  
 Garza, F., 178  
 Gasthaus, J., 62  
 Gastinger, J., 192  
 Gautier, P., 298  
 Gefang, D., 343  
 Ghomeshi, H., 169  
 Ghosh, J., 119, 304  
 Giannone, D., 348  
 Gilliland, M., 60, 83, 84, 374  
 Gimelshein, N., 291

- Giraud-Carrier, C.G., 223  
 Glorot, X., 64  
 Godahewa, R., 117, 119  
 Godinho, P., 342  
 Goldstein, D.G., 265  
 Gomes, H.M., 165, 170  
 Goncalves Jr, P. M., 169  
 Gonfalonieri, A., 21  
 Gönül, M.S., 266  
 Goodwin, P., 316, 322, 381  
 Gorodnichenko, Y., 342  
 Gorr, W.L., 229  
 Gouttes, A., 306  
 Grace, K., 43  
 Graepel, T., 334  
 Grah, R., 156  
 Granger, C.W.J., 51, 192  
 Grau, C., 22  
 Gray, S., 301  
 Grecov, P., 117  
 Greenwood, G.W., 223  
 Grewe, D., 334  
 Grigas, P., 317  
 Grosch, G., 116  
 Grose S., 53  
 Grushka-Cockayne, Y., 49, 51  
 Gruson, H., 156  
 Guestrin C., 253  
 Guez, A., 334  
 Günemann, S., 301, 302  
 Guns, T., 320  
 Guo, X., 316  
 Guo, Z.X., 316  
 Gupta, Y., 94  
 Gusset, F., 116
- H**  
 Ha, D., 223  
 Hadsell, R., 301  
 Hahmann, M., 258  
 Haider, A., 341, 342  
 Hall, H., 21  
 Hamoudia, M., 82, 83  
 Hanson, R., 36  
 Han, X., 119  
 Hao, W., 342  
 Happiette, M., 104  
 Harutyunyan, A., 335  
 Harwell, D., 39  
 Hasan, N., 342  
 Hassabis, D., 334  
 Hasson, H., 86  
 Hastie, T., 99  
 Hastie, T.J., 271  
 HosseiniFard, Z., 336  
 He, K., 291  
 Henderson, T., 229  
 Henighan, T., 301  
 Herrera, F., 182  
 Herzen, J., 116  
 Hewamalage, H., 110  
 Hielkrem, C., 84  
 Hill, S., 41  
 Hinterlang, N., 342  
 Hinton, G., 291  
 Hinton, G.E., 295  
 Hoffman, M.W., 223  
 Hoi, S., 168  
 Hospedales, T.M., 223  
 Huang, A., 334  
 Huang T., 58  
 Hubinger, E., 33  
 Huguenin, N., 116  
 Hu, J., 238  
 Hu, M.Y., 256  
 Hussein, A., 333  
 Hutter, M., 4  
 Hyde, R., 165, 166  
 Hyndman, R., 224, 229  
 Hyndman, R.J., 199, 202, 205, 225,  
                          375

**I**

- Ioffe, S., 64  
 Ivanov, S., 334

**J**

- Jaitly, N., 331  
 Januschowski, T., 54, 55, 228  
 Jayne, C., 333  
 Jeffrey, R., 88  
 Jenkins, G.M., 108  
 Jeon, Y., 304  
 Jha, N., 94  
 Jin, X., 303  
 Johnson, C., 41  
 Johnson, H., 156  
 Jones, B., 22  
 Jones, N.S., 260  
 Jordon, J., 297

**K**

- Kadlec, P., 223  
 Kalchbrenner, N., 334  
 Kallus, N., 321  
 Kalman, R.E., 53  
 Kang, P., 86  
 Kang, Y., 52, 53, 193, 253, 256–260,  
     262, 263, 266–268, 270, 271  
 Kan, K., 304  
 Kaplan, J., 301  
 Kapoor, S., 282  
 Kariniotakis, G., 60  
 Karnin, Z., 298  
 Kavukcuoglu, K., 334  
 Ke, G., 237, 245  
 Kegel, L., 258  
 Kempa-Liehr, A.W., 230  
 Kenett, D.Y., 342  
 Khalil, E., 326  
 Khandelwal, J., 304  
 Killeen, T., 291  
 Kim, B., 33

**Kim, S.G., 86**

- Kim, S.-L., 224  
 Kingma, D.P., 64  
 Kiros, J.R., 291  
 Knaute, P., 260  
 KoAcisz, J., 116  
 Koehler, A.B., 50  
 Kohn, R., 256  
 Kolassa, S., 53, 270  
 Kolter, J.Z., 165, 166, 170  
 Koltun, V., 62

**Kononenko, I., 264**

- Koren, M., 306  
 Kosina, L., 77  
 Kosinski, M., 116  
 Kossuth, J., 15  
 Kotary, J., 322  
 Kourentzes, N., 192, 265  
 Kovalchuk, Y., 168  
 Krawczyk, B., 169

**Kristjanpoller, W., 342**

- Krizhevsky, A., 291  
 Kropiński, P., 342  
 Kruger, E., 7  
 Kück M., 225  
 Kuhl, N., 165  
 Kumar, A., 168  
 Kumar, K., 53  
 Kundu, S., 344  
 Kunz, M., 88

**L**

- Lachapelle, S., 322  
 Lacoste-Julien, S., 322  
 Lai, G., 62  
 Lainder, A.D., 59, 69  
 Lamblin, P., 64  
 Lanctot, M., 334  
 Laptev, N., 281  
 Large J., 256  
 Larochelle, H., 223

- Lerrick, R.P., 265  
 Larsen, E., 322  
 Leach, M., 334  
 Leckie, C., 320  
 Le Cun, Y., 63  
 Lee, H., 86  
 Lee, K., 303  
 Lee, S.Y., 154  
 Legg, S., 4  
 Lehner W., 258  
 Lei Ba, J., 291  
 Lemke, C., 192, 225, 253  
 Le, Q.V., 223  
 Lerer, A., 291  
 Le Strat, Y., 256  
 Letham, B., 108  
 Liang, J., 318  
 Liberty, E., 298  
 Lichtendahl Jr., K.C., 265, 266  
 Liemert, P., 5, 17  
 Li, F., 202, 225, 256, 257  
 Li, J., 303  
 Li, K., 223  
 Li, L., 260  
 Lillicrap, T., 334  
 Li, M., 316  
 Lim, B., 114, 281, 301, 303  
 Lines J., 256  
 Lin, H., 307  
 Lin, Z., 291  
 Lipton, Z.C., 33  
 Li, S., 303  
 Little, M.A., 260  
 Liu, C., 168  
 Liu, H., 62  
 Liu, J., 223  
 Liu, T., 166  
 Liu, T.-Y., 245  
 Liu, Y. H., 68  
 Liu, Z., xxiii  
 Li, W.K., 256  
 Li, X., 253, 259–261, 268  
 Ljung, G.M., 108  
 Lodi, A., 334  
 Loeff, N., 114  
 Lofstead, J., 193  
 Loh, E., 304  
 Lolić, I., 342  
 Lombardi, M., 330  
 López-de Lacalle, J., 256  
 Lorenz, T., 39  
 Loriga, S., 342  
 Lotze, T.H., 256  
 Loureiro, A., 102  
 Louveaux, Q., 338  
 Lubba, C.H., 260  
 Lubman, D., 117  
 Lubman, D.I., 117  
 Ludermir, T.B., 224  
 Luellen, E., 94  
 Luengo, J., 179  
 Luna, J.M., 223  
 Lu, Q., 223  
 Lusk, E., 54  
 Lütkepohl, H., 52
- M**
- Mackenzie, D., 6  
 Macready W.G., 221  
 Maddison, C.J., 334  
 Maddix, D.C., 62  
 Madeka, D., 301  
 Magnus, J.R., 52  
 Makridakis, S., 43, 78, 79, 84, 87,  
     226, 245, 341, 379  
 Mak, T.W.K., 324  
 Ma, L., 88  
 Malik, J., 223  
 Mallick, B., 154  
 Malone, T., 26  
 Maloof, M.A., 165  
 Mandi, J., 317, 320  
 Mané, D., 33

- Mann, B., 301  
 Mannes, A.E., 265, 266  
 Ma, P., 223  
 Marcos Alvarez, A., 338  
 Marcucci, J., 342  
 Marcus, G., 6  
 Martinez-Costa, C., 103  
 Ma, S., 58  
 Mas-Machuca, M., 103  
 Massa, F., 291  
 Ma, W., 166  
 Mazyavkina, N., 334  
 McAfee, R.P., 265  
 McCandlish, S., 301  
 McGee, V.E., 84  
 McNellis, R., 318  
 Meade, N., 81, 85, 224  
 Medico, R., 304  
 Meldgaard, J.P., 59  
 Melnychuk, V., 306  
 Melo, P., 342  
 Meng, Q., 166, 168, 176, 237, 245,  
     300  
 Menon, S., 82  
 Mercado, P., 310  
 Mes, M.R.K., 91, 93, 94, 101  
 Micaelli, P., 223  
 Miguéis, V., 102  
 Mikolov, T., 288  
 Mikulik, V., 33  
 Milano, M., 330  
 Miley, J., 22  
 Miller, D.M., 52  
 Miller, J.D., 34, 41  
 Minku, L., 169  
 Mishchenko, K., 119  
 Moat, H.S., 342  
 Mohr, M., 165  
 Molina, M.D.M., 223  
 Montero-Manso, P., 225, 229  
 Montgomery J.B., 68  
 Montgomery, M., 119  
 Morlidge, S., 379  
 Morris, C., 334  
 Mozetic, I., 178  
 Moshiri, S., 341, 342  
 Mulamba, M., 319  
 Mulder, G., 59  
 Muller, V.C., 43  
 Musk, Elon, 8, 22
- N**  
 Naccarato, A., 342  
 Nagao, S., 342  
 Naghibi, T., 306  
 Nair, V., 295  
 Nallapati, R., 298  
 Narkhede, M.V., 197  
 Neuer, T., 116  
 Neuffer, J., 230  
 Nguyen, H.L., 166  
 Nham, J., 334  
 Nick, N., 343  
 Nicolas, S., 192  
 Niehus, R., 156  
 Nikolopoulos, K., 52  
 Novak, M., 13  
 Nowe, A., 335
- O**  
 Ogbonna, A.E., 342  
 O'Hara-Wild, M., 229  
 Olah, C., 33  
 Olivares, K.G., 168  
 Olson, M., 36  
 Önkal, D., 266  
 Oosterlee C.W., 62  
 Ord, K., 81  
 Orenstein, J., 41  
 Oreshkin, B.N., 61, 63, 113, 194,  
     202, 303

**P**

- Panagiotelis, A., 53  
 Panda, R.K., 342  
 Panko, B., 5  
 Park, H.W., 86  
 Pasieka, M., 116  
 Passerini, A., 335  
 Patel, Y., 301, 303  
 Patenaude J.M., 57  
 Pearl, J., 6, 287  
 Peersman, G., 343, 350  
 Peirano, R., 342  
 Peng, J., 303  
 Perrault, A., 319  
 Petitjean, F., 165, 166  
 Petrik, M., 116  
 Petropoulos, F., 49, 51–53, 60, 64,  
     192, 225, 226  
 Pfau, D., 223  
 Pfister, T., 114  
 Pham, H., 334  
 Piazzetta, S.G., 334  
 Pierini, A., 342  
 Pincheira, P., 342  
 Pinson, P., 60  
 Pinto, F., 225  
 Polat-Erdeniz, S., 330  
 Polemitis, A., xxiv  
 Pollock, A.C., 266  
 Popescu, A., 330  
 Popovici, D., 64  
 Pottelbergh, T.V., 116  
 Pramanik, N., 342  
 Prasanna, A.N., 117  
 Prasse, B., 156  
 Preis, T., 342  
 Proietti, T., 52  
 Prudêncio R.B., 224

**Q**

- Qiao-Feng, T., 342

Qi, M., 202

**R**

- Radford, A., 301  
 Raille, G., 116  
 Rajan, A., 94  
 Rajapaksha, D., 118  
 Ramamohanarao, K., 320  
 Raman, S., 381  
 Rangapuram, S., 113  
 Ranjan, R., 82  
 Ranzato, M., 307, 311  
 Raslan, M., 88  
 Rasul, K., 304, 306  
 Ravi, S., 223  
 Raymond, S.J., 68  
 Razbash, S.D., 59  
 Regan, B., 304  
 Reichlin, L., 348  
 Reif, M., 223  
 Reinsel, G.C., 108  
 Ren, S.Y., 291  
 Rice, J.R., 223, 252  
 Rickard, R., 376  
 Riise, T., 56  
 Robette, J., 374  
 Robnik-Šikonja, M., 264  
 Rolnick, D., 329  
 Romero, C., 223  
 Rouesnel, L., 298  
 Rubinoff, M., 250  
 Rus, D., 19  
 Ryder, N., 301

**S**

- Sadoughi, A., 298  
 Sahoo, D., 168  
 Sainz, M., 103  
 Salinas, D., 300, 303, 304  
 Salisu, A.A., 342  
 Salvatier, J., 43

- Sandberg, Anders, 32  
Sandman, F., 156  
Santos, S., 230  
Satzger, G., 165  
Savov, V., 24  
Say, J.B., 24  
Schaer, O., 103  
Schaul, T., 223  
Schelter, S., 282  
Schmidhuber, J., 223  
Schmidt, D., 119  
Schmidt, M., 192  
Schrittweisner, J., 334  
Schubert, S., 376  
Schulke, A., 192  
Schulman, J., 33  
Schultz, S.R., 260  
Schulz, J., 62  
Schuster, I., 304  
Scott, C., 21  
Scott, D., 117  
Seamans, R., 15  
Seeger, M.W., 113  
Semenoglou, A., 56, 59, 62, 68, 79, 89  
Sen, R., 114  
Seong, S., 304  
Šestanović, T., 341  
Sethi, S.S., 260  
Sglavo, U., 83  
Shafaei, M., 342  
Shafait, F., 223  
Shah, C., 224, 253  
Shah, S., 319  
Shanbhogue, R., 343  
Shang H.L., 56  
Sheikh, A., 304  
Shen, L., 238  
Shen, Y., 335  
Sherratt, K., 156  
Shih, S.Y., 62  
Shmueli, G., 256  
Shrestha, D.L., 60  
Siemsen, E., 49  
Sifre, L., 334  
Silver, D., 334  
Simchi-Levi, D., 253  
Singhania, R., 344  
Singh, H., 381  
Singh, P.K., 94  
Skalse, J., 33, 116  
Skrödski, A., 116  
Small, D., 348  
Smets, F., 343, 350  
Smirnov, P.S., 98  
Smith-Miles, K., 224  
Smyl, S., 56, 57, 63, 113, 202  
Snyder R.D., 53  
Soares, C., 225  
Sokele, M., 85  
Soll, J.B., 265  
Solomatine, D.P., 60  
Song, T.M., 88  
Sorić, P., 342  
Sorjamaa, A., 120  
Spiliotis, E., 53, 245  
Spithourakis, G.P., 52  
Srivastava, N., 291  
Stanley, H.E., 342  
Stefanowski, J., 169  
Steinhardt, J., 33  
Stella, L., 62  
Stempel, D., 343  
Stephan, J., 306  
Stepic, D., 192  
Sternberg, R.J., 4  
Stern, P., 86  
Storkey, A.J., 223  
Strachan, R., 343  
Stuckey, P.J., 320  
Subbiah, M., 301  
Sudakov, V.A., 98  
Sugiyama, M., 308  
Suilin, A., 113

Su, J., 342  
 Sun, F.K., 62  
 Sun, G., 238  
 Sun, J., 63  
 Sun, Y., 335  
 Suri, S., 265  
 Sutskever, I., 288, 303  
 Svetunkov, I., 53  
 Sviridov, S., 334  
 Syntetos, A., 322  
 Szafranek, K., 342, 344  
 Szczypula, J., 229  
 Szegedy, S., 64

**T**  
 Tafti, L., 116  
 Takeda, F., 342  
 Talagala, T., 225  
 Talagala, T.S., 225, 253, 258, 260, 268  
 Talavera, O., 342  
 Tambe, M., 319  
 Tanaka, R., 342  
 Tang, Y., 68  
 Tashman, L., 51, 68  
 Taylor, S.J., 108  
 Teräsvirta, T., 345  
 Teso, S., 335  
 Theil, H., 375  
 Theodorou, E., 259, 263, 264  
 Thomassey, S., 103, 104  
 Thomson, M.E., 265, 266  
 Tho, P., 342  
 Tibken, S., 12  
 Tibshirani, R., 99  
 Tichy, M., 379  
 Timmermann, A., 192  
 Tjozstheim, D., 56  
 Tkacz, G., 341  
 Torgo, L., 225  
 Torkkola, K., 54

Toutanova, K., 303  
 Tran, C., 329  
 Trapero, J.R., 112  
 Trichy, V.K., 86  
 Tsymbal, A., 166  
 Tung, L., 14  
 Turkmen, A.C., 62

**U**  
 Udenio, M., 57  
 Urban, T., 22  
 Urquhart, A., 342  
 Uta, M., 330

**V**  
 Vaggi, F., 119  
 Van den Driessche, G., 334  
 Van der Schaar, M., 297  
 Van Hentenryck, P., 324  
 van Merwijk, C., 33  
 Van Pelt, J., 342  
 Van Steenbergen, R.M., 91  
 Vanston, L., 81, 82, 85, 86  
 Vardnman, R., 29  
 Varian, H., 343  
 Vasnev, A.L., 52  
 Ventura, S., 223  
 Villani, M., 256  
 Vinod, H.D., 256  
 Vinyals, O., 331  
 Vollgraf, R., 304

**W**  
 Wagner, J., 22  
 Wang, E., 229  
 Wang, J., 223  
 Wang, J.X., 223  
 Wang, L., 342  
 Wang, S., 57  
 Wang, T., 166

- Wang, W., 52  
 Wang, X., 192, 224, 253, 260,  
     265–269, 271  
 Wang, Y., 62  
 Wang, Y.X., 303  
 Wang, Z., 223  
 Wan, L., 63  
 Warren, T., 25  
 Webb, G.I., 165, 166  
 Wehenkel, L., 338  
 Weise, C.L., 343  
 Wellens, A.P., 57, 68  
 Werner, L.D., 310  
 Widmer, G., 169  
 Wilder, B., 318, 322  
 Williams, D., 52  
 Williams, J.R., 68  
 Williamson, C., 116  
 Wilson, E., 381  
 Winkler, R.L., 265  
 Wolfinger, R.D., 59  
 Wolpert, D.H., 221, 251  
 Wong, C.S., 256, 257  
 Woo, G., 168  
 Wright, M.J., 86  
 Wu, E., 238  
 Wu, J., 301
- X**  
 Xiang, B., 298  
 Xiao-Hui, L., 342  
 Xin, W., 342  
 Xiong, H., 303  
 Xuan, Y., 303  
 Xu, W., 342
- Y**  
 Yampolskiy, R.V., 33, 34  
 Yanfei, K., xxvi  
 Yang, K., 68  
 Yang, Y., 62  
 Yasmeen, F., 53  
 Ye, Q., 166  
 Ye, R., 168  
 Yi, J., 342  
 Yosinsk, J., 281  
 Yovits, M.C., 250  
 Yudkowsky, E., 35, 41, 42  
 Yu, H., 266  
 Yu, H.F., 114
- Z**  
 Zahner, J., 343  
 Zaniolo, C., 169  
 Zaucha, J., 41  
 Zeiler, M., 63  
 Zhang, B., 43  
 Zhang, G., 54  
 Zhang, G.P., 202, 256, 342  
 Zhang, S., 68  
 Zhang, W., 223  
 Zhang, X., 63  
 Zhang, Y., 334  
 Zhang, Z., 68  
 Zhou, H., 303  
 Zhou, X., 303  
 Zico Kolter, J., 321, 323  
 Ziel, F., 49  
 Zimmermann, K.F., 342  
 Zou, H., 342

## SUBJECT INDEX

### A

- Academic, 34, 86, 97  
Achievements, 4, 7, 8, 12, 15, 16, 24, 269  
Adam optimizers, 63, 245  
Adaptive Boosting (AdaBoost), xxxi  
Adaptive Boosting Model, 169  
Adaptive Dynamic Programming (ADP), xxxi  
Adaptive Fuzzy-Neural-Network Controller (AFNNC), xxxi  
Adaptive Neural Control (ANC), xxxi  
Additions to Existing Product Lines, 80  
Additive seasonality, 50  
Adjusted, 52, 60, 175, 182, 198, 350, 381  
ADL with data Pooling (ADLP), 244  
Adoption, 81, 85, 86, 193, 195, 382  
Advancements, 32, 33, 35, 43, 107, 252, 373  
Advancements in AI technology, 252  
Advantage, 4, 5, 7, 19, 20, 23, 37–40, 51, 54, 57, 61, 78, 87, 98, 195, 281, 282, 297, 300, 344, 348  
Agglomeration, 139  
Aggregation, 52, 53, 257, 344, 348, 363, 366, 367  
AI development, 35, 38–40  
Alexa, 9, 12, 14, 19  
Algorithms, 5, 6, 18, 23, 33, 53–56, 63, 87, 89–92, 96, 114, 169, 170, 173, 197, 198, 223, 230, 241, 252, 256, 258, 259, 262, 263, 265–267, 271, 272, 323, 326–332, 334, 341, 349, 363  
Alphabet, 17, 38  
AlphaGoZero, 11  
AlphaStar, 11, 18  
Amazon, 12, 14, 298  
American Heritage Dictionary, 4  
Analogs, 79, 81, 90  
Analogous, 3, 81, 86, 97, 98, 101, 257  
Analytics, 316  
Anticipation, 353  
Apple, 23, 38, 80

- Application Programming Interface (API), 116, 203
- Apps in mobile phones, 11, 12
- Architects of Intelligence, 15, 16
- ARIMA-ANN model, 342
- Artificial General Intelligence (AGI), 16
- Artificial Intelligence (AI), 3–8, 14–16, 18–20, 23–26, 32–44, 123–126, 160, 161, 223, 251, 271, 373
- Artificial Neural Networks (ANN), 86, 88, 89, 100, 224, 341–345, 347, 356, 365
- Artificial Super Intelligence (ASI), xxxi
- Assumptions, 31, 42, 50, 53, 54, 56, 58, 124, 125, 136, 155, 193, 279, 280, 295, 344, 346, 347, 366
- Attribute, 51, 81, 89, 95, 97, 98, 258, 264, 305
- Attributes of item, 95
- Augmented Reality (AR), xxxi
- Australia, 39, 40, 156
- Autocorrelation, 259, 260
- AutoKeras, 59
- Automate, 13, 35, 59, 98
- automatic Machine Learning (aML), xxxi
- Automatic modeling, 377
- Automatic model selection, 377
- Automatic Speech Translation, 12
- Automatic Text Translation, 11
- Automatic translations, 9
- Automation of feature extraction, 255, 260, 272
- AutoML, 59
- Autonomous Vehicles (AVs), 4, 10, 12
- Autoregression, 57, 128, 133, 141, 147, 148, 152, 350
- Auto Regressive (AR), xxxi
- Autoregressive cross learning models, 161
- Autoregressive Distributed Lag (ADL), 244
- Autoregressive Integrated Moving Average (ARIMA), 50, 53, 55–57, 82, 116, 124, 152, 229, 244, 258, 342, 362
- Autoregressive integrated moving average with external variables (ARIMAX), 58
- Average Prediction Error (APE), xxxi
- Average Relative MAE (AveRelMAE), 381
- B**
- Backpropagation, 58, 295
- Back-testing, 204, 215
- Baidu's Apolong, 12
- Balance, 38, 39, 85, 200
- Baseline model, 179, 187, 196, 203
- Bass model, 78, 85
- Batch Mode Active Learning (BMAL), xxxii
- Bayesian Belief Network (BBN), xxxii
- Bayesian Models (BMs), xxxii
- Bayesian Network (BN), xxxii
- Bayesian Neural Network (BNN), xxxii
- Benchmarks, 63, 92–94, 99–101, 166, 176, 178, 179, 184, 226, 232, 255, 343, 379
- Benefit, 18, 36, 40, 50, 51, 56, 79, 100, 127, 135, 153, 158, 161, 195, 200, 201, 203, 205, 206, 225, 266, 315, 316, 321, 325, 326, 328, 330, 366, 380, 382
- Best-performing model, 245, 253, 362
- Bidirectional Recurrent Neural Network (BRNN), xxxii

- Big Data, 9, 57, 107, 108, 222, 246, 271  
 Big Tech, 36  
 Black Box of Machine Learning, 33  
 Black Price, 284, 285, 294, 295, 299  
 Black Swans, 10  
 Boolean Control Networks (BCNs), xxxi  
 Bootstrapping, 52–54, 69  
 Bounded Component Analysis (BCA), xxxi  
 BrainNet, 22  
 Brain-to-Brain Interfaces (BBIs), 22  
 Business Cycle, 345, 365  
 Business value, 374
- C**  
 Calculation, 95, 165, 173, 230, 263, 272, 349  
 California, 17, 352  
 Cannibalization, 95  
 Capabilities, 4, 8, 19, 21, 25, 33, 35, 88, 107, 341, 344, 354, 364, 365  
 Capacity, 33, 36, 38, 54, 55, 80, 133, 135, 334  
 CatBoost (CB), 87, 96, 98, 99  
 Category, 14, 79, 80, 84, 93, 98, 99, 234, 300  
 Causal, 6, 288, 294, 297, 306  
 Causal inference, 24, 117  
 CB Insights, 15  
 Challenges, 4, 8, 15, 33, 42, 51, 87, 89, 118, 193, 214, 223, 279–281, 284, 304, 333  
 Change(s), 5, 7–10, 19, 32, 33, 35, 37, 41–43, 83–85, 97, 99, 101, 197, 214, 257, 259, 265, 374, 378, 381, 382  
 ChatGPT, 24, 25, 32–34, 43, 88  
 Chess and Shogi, 11, 18  
 China, 36, 38–40, 43, 154  
 Classical time series models, 160  
 Classification And Regression Trees (CARTs), 327  
 Classifier, 98, 169, 170, 225, 258  
 Clients, 43  
 Cloud, 9, 21  
 Collinear, 59  
 Combination of forecasting methods, xii  
 Companies, 15, 17, 21, 22, 38, 40–42, 60, 92, 93, 101, 107, 301, 353, 356, 379  
 Competition, 14, 20, 25, 36, 38, 40, 52, 56, 62, 77, 78, 80, 83, 87, 110, 112, 113, 224, 381, 382  
 Complex Dynamical Networks (CDNs), xxxii  
 Component, 33, 50, 52, 136, 138, 150, 222, 224, 256, 257, 352  
 Computational cost, 50, 51, 60, 61, 225  
 Computational scalability, 99  
 Computer programming, 35  
 Computers, 14, 18, 20–26, 223  
 Computing power, 32  
 Concentrated, 36  
 Concept drift handling, 165–172, 179, 186  
 Concept drift types, 166, 167, 170, 171, 176, 177, 179–184, 186, 187  
 Concern, 8, 14–17, 32, 34, 39, 40, 42, 44, 87, 222, 378  
 Consensus of forecasts, 101  
 Consensus of Sales Team, 84  
 Consequences, 32, 38, 39, 44, 126, 134  
 Constructing ensembles, 192, 194  
 Consumer adoptions, 81, 85  
 Content, 34  
 Continuous Action-set Learning Automata (CALA), xxxii

- Contribution, 39, 195, 223, 280, 306, 326
- Controllability, 33
- Conventional stochastic programming methods, 322
- Convolutional blocks, 238, 239, 241, 245
- Convolutional filters, 245
- Convolutional neural network (CNN), 88, 97, 226, 230, 246, 260, 334
- Coordinate Descent (CD), xxxii
- Cornell University, 22
- Correlation, 57, 62, 228, 260, 264, 266, 289, 293, 314, 319, 343, 365
- Cost, 13, 15, 17, 37, 39, 59, 60, 88, 89, 100, 124, 151, 195, 204, 215, 222, 325
- Covariates, 86, 128, 281, 284–289, 292, 304
- COVID, 127, 153–155, 157–159, 161, 342, 354
- Cross-learn Artificial Intelligence models, 160
- Cross-learning, 56, 78, 79, 124–127, 130–142, 145, 147, 150–153, 155, 156, 158, 160, 161, 198, 202, 203
- Cross-learning equivalence, 161
- Cross-learning mechanism, 140, 160
- Cross-learning modeling, 130
- Cross-sectional relationships, 50, 64
- Cultural differences, 160
- Customer Confidence, 344, 345, 364
- Customers surveys, 84
- Customised loss function, 317
- D**
- Data, 5, 6, 20, 22, 25, 33, 37, 78–81, 85–93, 95–101, 107–114, 117, 119, 123–126, 132–134, 136, 138, 141, 144, 147, 151–160, 163, 164, 166–170, 172, 177–179, 185–187, 193, 194, 200–202, 208, 214, 222, 224–226, 228–230, 232, 234–237, 244–246, 280, 282, 284, 290, 292, 297, 376–383
- Data augmentation, 52, 59, 68, 117
- Data augmentation techniques, 68, 69
- Data-driven forecasting, 280
- DataFrame, 232, 234
- Data generating processes (DGPs), 50, 51, 54, 66, 132, 258
- Data generation, 253, 255
- Data generation process, 176, 187, 257
- Data partitioning, 115
- Data pooling, 124, 130, 244
- Data processing, 52, 59, 63, 65, 230
- Data scientist, 59, 373, 377, 379
- Data set, 50, 55, 57, 62, 88, 90, 92–94, 96, 98–101, 199, 201, 213
- DB-Scan, 115
- Decision-error function, 317
- Decision-making, 32, 44, 107, 135, 269, 314, 316, 322, 323, 325, 327
- Decision-making under uncertainty, 321
- Decision Support System (DSS), xxxii
- Decomposition, 202, 224, 260
- DeepAR, 88, 113, 116, 194
- Deep AutoRegressive RNN, 88
- Deep Belief Network (DBN), xxxii
- Deep Constraint Completion and Correction (DC3), 329
- Deep Convolutional Generative Adversarial Network (DCGAN), xxxiii
- Deep Learning architectures, 329

- Deep learning (DL), 6, 14, 18, 23, 50, 79, 88, 89, 98, 116, 168, 176, 196, 203, 231  
 Deep learning methods, 61  
 DeepTime, 168  
 Delayed Memristive Neural Networks (DMNNs), xxxii  
 Delayed Recurrent Neural Networks (DRNNs), xxxii  
 Delphi method, 78, 83  
 Demand error, 88, 299–302  
 Demand forecasting, 78, 81, 87, 89, 100, 229, 282, 287, 292, 297, 305, 306  
 Demand Forecast model, 287, 288, 293, 297  
 DemandForest, 91–94  
 Demand pattern, 92  
*Demand Profiles of New Products*, 91  
 Derived features, 95  
 Develop, 6, 21, 23–26, 33, 36, 38, 43, 55, 56, 197, 200, 203, 213, 222, 316, 321, 324, 365  
 Development, 23, 33–38, 40, 43, 49, 53, 89, 99, 100, 222, 246, 303, 366  
 Diagnosis, 36, 88  
 Diffusion models, 78, 85, 90  
 Dimensionality reduction, 225  
 Direct Brain to Computer Interface (BCI), 21–23  
 Discount, 32, 95, 99, 283, 285, 286, 288, 294, 297  
 Discrete-time Neural Networks (DNNs), xxxii  
 Discriminant analysis, 224, 253  
 Disruption, 19, 32  
 Distribution, 39, 88, 91, 92, 132, 151, 163–166, 185, 205, 210, 244, 256, 257, 264, 282, 285, 382, 383  
 Dragon Home 15, 11  
 DSGE model, 343  
 Dynamic-global, 284
- E**  
 e-commerce, 100  
 Economic growth, 36, 43  
 Economies of scale, 37–40, 94  
 ECross, 136  
 Education, 25, 31, 32, 34  
 Effects of interventions, 158  
 Efficiencies, 37  
 EIA, 165, 170, 171  
 ELocal, 136  
 Emerge, 35, 88, 92, 165, 213, 253  
 Emphasis, 35, 296  
 Employment, 25, 35, 344, 350, 353, 354, 356  
 Ensemble, 51, 63, 87, 192–196, 198, 200–204, 208, 210, 222, 225, 228, 239, 240, 256  
 Ensemble empirical mode decomposition EEMD-ANN, 342  
 Entropy, 229, 256, 258, 260, 264, 375  
 Error-based combinations, 66  
 Error Contribution Weighting (ECW), 165, 173  
 Error Interaction Approach (EIA), 165, 170, 171  
 Error metrics, 166, 176–178, 181, 185  
 Error Trend and Seasonality (ETS), 118, 164, 171, 179  
 Error variability, 192  
 Estimation Error Covariance (EEC), xxxii  
 Europe, 39, 153, 154, 280  
 Evaluation, 99, 116, 178, 241, 284, 295, 335, 376, 381–383  
 Evolutionary Artificial Neural Networks (EANNs), xxxii  
 Executive review, 377, 380, 381

- Expectations, 8, 12, 16, 31–38, 41–43, 69, 77, 78, 84  
 Experimental design, 195, 196, 201  
 Expert opinion, 78, 83, 90  
 Expert opinion methods, 83, 84  
 Experts' behaviour, 333  
 Explanatory variables, 50, 61, 65, 87, 228  
 Exponential growth, 142, 161  
 Exponential Smoothing (ETS), 50, 53, 58, 124, 134, 152, 164, 229, 244, 245, 342, 362, 379  
 Exponential Smoothing-Recurrent Neural Network (ES-RNN), 113  
 Extreme learning machine (ELM), 244
- F**  
 Facebook, 34, 39  
 FaceNet, 11  
 Face recognition, 5, 9, 11, 15, 19  
 Fashion retail, 88, 94  
 Feature, 38, 80, 84, 87, 89, 94–96, 98, 99, 101, 108, 167, 224–226, 228–232, 234–236, 238, 239, 241, 244, 245, 375, 377, 382  
 Feature-based Forecast Model  
     Averaging (FFORMA), 226, 253  
 Feature-based FOREcast Model  
     Selection (FFORMS), 225, 226  
 Feature-based methods, 252, 269, 271  
 Feature-based model combination, xii  
 Feature-based model selection, xii  
 Feature extraction, 229, 230, 238, 255, 258  
 Financial markets, 31, 42  
 Firms, 14, 15, 17, 37, 38, 40, 43, 44  
 Fitting, 53, 55, 111, 114, 193  
 Forecastability, 375  
 Forecast accuracy, 87, 98, 118, 134, 299, 305, 316, 317, 374–376, 378  
 Forecast combination, 51, 52, 165, 226, 253, 260, 262–264, 342, 382  
 Forecast cycle, 65  
 Forecast diversity, 255, 262, 263, 272  
 Forecast error, 52, 54, 60, 78, 126, 198, 225, 272, 324  
 Forecast explainability, 50, 66  
 Forecasting, 6, 43, 77–83, 86–91, 94, 99–101, 221, 222, 224, 225, 228–231, 234–237, 244–247, 251, 253, 255, 260, 269, 270, 374–377, 379–381, 383  
 Forecasting accuracy, 50, 60, 78, 125, 161, 163, 182, 192, 197, 203–206, 208, 212, 214, 224, 231, 263, 281, 304, 314, 351, 361  
 Forecasting approach, 50, 58, 62, 64, 108, 201, 287, 301  
 Forecasting errors, 185, 205, 209, 210, 225, 241, 253  
 Forecasting function, 132, 234  
 Forecasting horizon, 54, 63, 90, 193, 194, 200–202, 210, 234–236, 244, 270, 271  
 Forecasting large collections of time series, 252  
 Forecasting methods, 50, 51, 56, 57, 79, 98, 101, 112, 164, 165, 191–193, 195, 224, 225, 228–230, 253, 268, 270  
 Forecasting performance, 53, 59, 60, 89, 136, 193, 194, 196, 198, 200, 201, 203, 206, 208, 210, 224–226, 228–231, 240, 252, 272, 362, 363, 365, 373, 374, 377, 378, 380, 383  
 Forecasting pool, 229, 255, 263, 270

Forecast trimming, 255, 264, 266  
 Forecast value added (FVA), 373–383  
 Frequency, 52, 124, 142, 145, 200,  
 202, 257, 291

Future Prospects, 16  
 Fuzzy approaches, 140

## G

Games, 3, 5–8, 18, 20, 24, 333  
 Gaussian mixture autoregressive  
 (MAR), 256  
 GDP growth, 123  
 Generalized Additive Model, 254  
 Generalized Bass model, 86  
 Generalized Regression Neural  
 Network (GRNN), xxxii  
 GeneRAting TIme Series, 256, 258  
 Generic forecasting solutions, 53  
 Germany, 38, 300  
 GitHub, 24  
 GitHub’s Copilot, 24  
 Global economy, 38, 40  
 Global Forecasting Models (GFMs),  
 108, 164, 228, 285  
 Global Models, 56, 57, 59, 109,  
 111–114, 116, 168, 245  
 Global Time Series Forecasting, 167  
 Global training strategies, 56  
 Go, 6, 9, 11, 18  
 Gompertz curve, 85  
 Goods, 37, 39, 77, 98, 99, 376  
 Google, 8, 12, 14, 21, 23, 24, 113,  
 343  
 Google assistant, 9  
 Google’s Tensorflow API, 203  
 Government, 42, 43  
 Gradient Boosted Trees (GBTs), 86,  
 87, 89, 98–100, 164  
 Gradient Boosting Regression Trees  
 (GBRTs), 244  
 Gradient Descent Weighting (GDW),  
 165, 174

Gradients, 63, 87, 98, 99, 174, 175,  
 231, 236

## H

Handwriting, 11  
 Hardware, 37, 38  
 Hedgehogs, 4, 18, 19  
 Heterogeneous time series, 131  
 Hierarchical Clustering Analysis  
 (HCA), xxxii  
 Hierarchical Reinforcement Learning  
 (HRL), xxxii  
 High-quality, 34  
 Human, 5–7, 11, 13, 14, 16, 17, 20,  
 23, 25, 32–35, 43, 51, 100, 222,  
 259, 333  
 Human intelligence (HI), 3–5, 7, 8,  
 12, 15, 19, 20, 24, 26, 36  
 Hybrid model, 342, 343  
 Hyperconnectivity, 26  
 Hyperparameters, 55, 59, 87, 89,  
 179, 334  
 Hyperparameter selection, 377

## I

IBM, 22, 24  
 Identifying, 9, 15, 23, 44, 51, 53, 55,  
 61, 89, 90, 192, 198, 209, 214,  
 230, 361, 374, 383  
 Image Classification, 11  
 Imagenet, 11, 261  
 Image recognition, 3, 18, 88, 124,  
 223  
 Images, 24, 32, 62, 88, 97, 99, 124,  
 260  
 Implementation, 8, 53, 63, 97, 100,  
 116, 166, 195, 201, 247  
 Implications, 3, 8, 15, 16, 18, 19, 21,  
 25, 26, 44, 269  
 Improvement, 7, 12, 15, 18, 20, 32,  
 35, 36, 78, 80, 86, 93, 94, 100,

- 182, 192, 194, 197, 206, 208, 211–213, 301, 315, 342, 373, 374, 381–383
- Impulse Response Function (IRF), 345, 349, 360
- Independent and Identically Distributed, 151
- Independent sample, 56, 132
- India, 40
- Industrial applications, ix
- Industry, 8, 37–40, 43, 83, 86, 92, 100, 201, 279–281, 352
- Inflation forecasting, 341, 344
- Information-criteria-based weighting, 66
- Information Theoretic Learning (ITL), xxxii
- Infrastructure, 37, 154
- Initialization, 63, 64, 67, 194, 204, 223, 232
- In-sample and out-of-sample, 66
- Inscrutable, 25, 26
- Intelligence Augmentation (IA), 4, 19–21, 23, 25, 26
- Interdependence, 111
- Interest, 37, 42, 60, 61, 95, 100, 118, 129, 223, 246, 258
- Internet, 11, 22, 342
- Internet traffic, 94
- Interpretability, 100, 135, 140, 225, 301
- Intersection of machine learning with forecasting and optimisation, 313–336
- Invasive Forms of Interfaces, 22
- Inventory, 91, 101, 279, 315, 383
- Inventory assumption, 280
- Inventory control, 322
- Investments, 15, 38, 41, 78
- iRobot*, 15
- IT, 269
- Iteration, 58, 113
- J
- Japan, 39
- Job creation, 35
- Job destruction, 35
- Judgmental methods, 78
- Judgmental model selection, 377
- Judgmental overrides, 374, 376, 377
- K
- Kernel Recursive Least-Squares (KRLS), xxxii
- Key Performance Indicator (KPI), xxxii
- K-means, 89, 91, 92, 115, 261
- K-Nearest Neighbors Regression (KNNR), 54
- Krafcik, 12
- Kurzweil, 7, 16
- L
- Labor markets, 32, 35, 41
- Lags, 57, 115, 128, 147, 236, 244, 245, 352
- LaMDA bot, 24
- Landscape, 41, 303
- Laplacian Support Vector Machine (LapSVM), xxxiii
- Large-scale, 37, 38, 119, 222, 231, 329
- Large-scale time series forecasting, 226, 228
- LASSO regression, 344
- Layers, 61–63, 88, 150, 194, 203, 213, 239, 344, 347, 376
- Learn how to learn, 230
- Learning capacity, 54, 55
- Learning rate, 63, 87, 179, 245
- Learning to learn, 223
- Least squares estimation, 133
- Least-Squares Support Vector Machines (LSSVMs), xxxiii

- Libratus, 11
- LightGBM (LGBM), 87, 166, 168, 176, 178, 179, 181, 300
- Light Gradient Machine (LGBM), 96
- Likelihood, 41, 44, 57, 84
- Limitations, 7, 8, 12, 23, 33, 54, 59, 137, 140, 169, 231, 281, 381
- Linear auto-regressive GFM, 112
- Linear optimisation, 324
- Linear weighting, 165, 171, 175, 178, 179, 181
- Links, 6, 21, 228, 268
- Local and cross-learning approaches, 131
- Locality and globality, 127, 130
- Locality-constrained linear coding (LLC), 261
- Local linear autoregressive model, 159
- Locally Optimised Decision Loss (LODL), 319
- Local model, 56, 57, 125, 132, 137, 245
- Local modeling, 130
- Logistic curve, 141
- Logistic growth, 126
- Logistic Smooth Transition Vector AutoRegressive (LSTVAR), 343
- Long Short-Term Memory (LSTM), 62, 88, 194
- Long-term investment, 34
- Loss criterion, 241
- Loss function, 60, 63, 87, 95–97, 133, 172–174, 193, 194, 196, 198, 200, 208, 209, 211, 231, 270, 296, 316
- Loss prediction, 230, 231, 235, 241
- Low sample problem, 124
- LSTVAR-ANN hybrid model, 343–345, 351, 355, 364
- LSTVAR model, 343, 345–347, 349, 366, 367
- M**
- M4 competitions, 56, 79, 226, 268, 270
- M5 competitions, 79, 86, 87, 90, 244, 263
- M4 dataset, 168, 255
- M5 forecasting competition, 108–110, 113, 115
- Machine learning (ML), 6, 33, 50, 54, 78, 82, 90, 113, 114, 116, 223, 231, 246, 373, 377, 382
- Magnitude warping, 69
- Marcus/Davies, 6
- Market, 15, 24, 34, 37–39, 41, 42, 78–80, 83–86, 90, 93, 98, 99, 280, 281, 342, 379
- Marketing review, 381
- Market penetration, 81, 83, 85
- Market research, 78, 83, 85
- Markov Decision Processes (MDPs), 332
- Markov process, 333
- Maximum likelihood estimation (MLE), 57
- M competitions, 113, 379, 382
- Mean Absolute Error (MAE), 177–179, 185, 198, 245, 299, 375, 381
- Mean Absolute Percentage Error (MAPE), 208, 209, 211, 212, 299, 300, 375, 378
- Mean Absolute Scaled Error (MASE), 375, 381
- Mean Squared Error (MSE), 96, 97, 198, 245, 266
- Measurement, 124, 225, 342
- Medium-term, 34, 107
- Memristor-based Cellular Neural Networks (MCNNs), xxiii
- Memristor-based Neural Networks (MNNs), xxix

- Memristor-based Recurrent Neural Networks (MRNNs), [xxxiii](#)
- Merchandising factors, [95](#)
- Merchandising variables, [95](#)
- MetaComb, [238](#), [239](#), [244–246](#)
- MetaCombFeat, [239](#), [240](#)
- Meta-combination, [238](#), [239](#)
- Metadata, [135](#), [223](#), [228](#), [229](#), [232](#), [235](#), [236](#), [238](#), [239](#), [241](#), [242](#), [246](#)
- Meta-forecasting, [222](#), [226](#), [228](#), [231](#), [240](#), [241](#)
- Meta Global-Local Auto-Regression (Meta-GLAR), [168](#)
- Meta-heuristic algorithms, [330](#)
- Meta-knowledge, [222](#), [228](#)
- Meta-learners, [68](#), [192](#), [224–226](#), [228–232](#), [239–242](#), [244–246](#), [253](#), [268](#), [271](#), [272](#)
- Meta-learning, [222–226](#), [228–232](#), [238](#), [245](#), [246](#), [252](#), [253](#)
- Meta-learning framework, [168](#), [222](#), [225–227](#), [246](#), [247](#), [253](#)
- MetaLoss, [241](#), [245](#), [246](#)
- MetaMoji, [11](#)
- Meta-selection, [231](#)
- Microeconomics, [32](#), [79](#)
- Microsoft, [38](#)
- Military, [38–41](#), [43](#)
- Mind/body/machine interface (MBMI), [23](#)
- Minimizing a loss function, [133](#)
- Missing values, [98](#)
- Mixed Integer Programming (MIP), [317](#)
- MobileEye, [8](#)
- Mobile phone, [80](#)
- Model Architecture, [116](#), [288](#), [290](#), [295](#)
- Model benchmarking, [299](#)
- Model class, [116](#), [129](#)
- Model class complexity, [136](#)
- Model classes for cross-learning, [152](#)
- Model complexity, [114](#), [137](#), [150](#), [153](#)
- Model driven, [50](#)
- Model-Free Adaptive Control (MFAC), [xxxiii](#)
- Modeling strategies, [228](#)
- Model Interpretability, [118](#)
- Model performance, [170](#), [178](#), [209](#), [281](#), [351](#)
- Monetary impact, [374](#)
- Monetary policy, [342–344](#), [350](#), [351](#), [361](#), [364](#)
- Money, [31](#), [32](#), [34](#), [37](#), [41](#), [42](#), [344](#)
- Monotonic Demand Layer, [289](#), [293–295](#), [298](#)
- Monte Carlo simulations, [53](#)
- Monthly, [195](#), [271](#), [350](#)
- Moore’s Law, [23](#), [86](#)
- Motivation, [126](#), [150](#), [153](#)
- Multidimensional Recurrent Neural Network (MDRNN), [xxxiii](#)
- Multi-Input Multi-Output (MIMO), [111](#)
- Multi-Layer Perceptron (MLP), [96](#), [194](#), [203](#)
- Multiple Extended Kalman Algorithm (MEKA), [xxxiii](#)
- Multiple temporal aggregation, [52](#), [53](#), [67](#)
- Multiple Timescale Recurrent Neural Network (MTRNN), [xxxiii](#)
- Multiple time varying variables, [111](#)
- Multi-step forecasting process, [378](#)
- Multivariate Adaptive Regression Splines (MARS), [xxxiii](#)
- Multivariate surface regression approach, [225](#)
- Multivariate techniques, [78](#)

**N**

Naïve benchmark, [96](#), [99](#)

- Naïve forecast, 374–377, 379, 381, 383  
 Nanotechnology, 4, 23  
 Natural Language Processing (NLP), 20, 114, 123, 301, 303  
 N-BEATS, 61–63, 168, 194  
 Netflix, 12  
 Neural Basis Expansion Analysis for Time Series (N-BEATS), xxxiii  
 Neural Coefficient Smooth Transition Autoregressive (NCSTAR), xxxiii  
 Neuralink, 22  
 Neural Network (NN), 15, 54, 97, 192, 193, 206, 225, 230–232, 238, 239, 246, 253, 256  
 Neural Turing Machine (NTM), xxxiii  
 Neuroscience, 4, 23  
 New product, 57, 77–81, 84, 86, 87, 89–94, 98–101, 258  
 New-to-the-firm products, 80  
 New-to-the-world, 79, 80, 90  
 Noise, 6, 8–10, 53, 69, 132, 151, 152, 166, 229, 256, 261, 319, 352  
 Non-Invasive Forms of Interfaces, 21  
 Non-invasive mind/body/machine interface (MBMI), 23  
 Non-invasive technology, 21  
 Non-linear Autoregressive Moving Average (NARMA), xxxiii  
 Non-linearities, 133  
 Nonlinear relationships, 50, 65  
 Nonlinear statistical models, 53, 88  
 Non monotonic, 147  
 Non-parametric Friedman rank-sum test, 179  
 Nonstationary, 53, 59, 256, 257  
 Normalization, 64, 148, 202, 293  
 Normally distributed forecast errors, 50  
 Notes Plus, 11  
 Nutonomy, 8
- O**  
 Objective, 5, 8, 11, 13, 14, 23, 33, 50, 52, 59, 64, 81, 84, 98, 109, 198, 230, 314, 316, 320, 377, 380  
 Obsolescence, 36, 270  
 Obsolete, 20, 34–36, 333  
 OneP forecast for each period, 92  
 Online fashion industry, 280  
 OpenAI's, 24, 32  
 Operations, 37, 238, 291, 314, 315, 324  
 Opportunities, 35, 49, 98, 118, 315, 354, 383  
 Optimal clusters, 92  
 Optimisation models, 314, 315, 318, 322–328, 330  
 Optimization methods, 55  
 Outcomes, 33, 39, 43, 44, 269  
 Outliers, 99, 299, 319  
 Out-thinking, 33  
 Overfitting, 51, 59, 63, 67, 99, 114, 239  
 Overidentification, 135  
 Overpredicting, 101
- P**  
 Paradigm, 124, 140, 153, 160, 314, 315, 336  
 Parameterization, 141, 142, 146, 154, 294, 295  
 Parameters, 85, 86, 90, 95, 101, 123, 124, 131, 150, 154, 223, 239, 241, 298, 304, 314, 321, 327, 333  
 Pattern of adoption, 81  
 Pattern recognition, 54, 88  
 Periodic patterns, 126, 140  
 Personal assistants, 9, 12, 14  
 Pharmaceutical evaluation, 376  
 Placebo, 376, 379

- Planning, 44, 49, 78, 107, 163, 269, 315, 322, 324, 383  
 Player, 11, 39, 40  
 Poisson, 96  
 Polynomials, 142, 145, 150, 161  
 Pooled Regression models, 112, 115  
 Pooling, 111, 112, 239, 261  
 Positional encoding, 289, 291  
 Possible Minds, 15, 16  
 Potential impacts, 33  
 Potential market, 83  
 Practitioners, 50, 56, 86, 108, 113, 116, 118, 191, 196, 198, 222, 232, 258, 314, 315, 336  
 Predicting, 24, 33, 43, 78, 99, 132, 164, 222, 290, 296, 314, 320  
*Predicting new product demand*, 86  
 Prediction algorithms, 114  
 Predictive accuracy, 100, 124, 301, 322  
 Predictive analytics, 313, 316  
 Predictive function, 125, 128, 131  
 Predictive performance, 94, 153, 244, 301  
 Prelaunch forecasts, 91  
 Prescriptive analytics, 313, 316  
 Principal Component Analysis (PCA), 225, 344  
 Probabilistic forecasting, 53, 62, 304, 324  
 Probabilistic forecasts, 53, 54, 60, 61, 113, 269, 382  
 Probability, 16, 53, 97, 151, 177, 240, 282, 285, 287  
 Process, 51, 55, 56, 78, 98, 99, 230, 239, 374–383  
 Production, 37, 39, 83, 94, 281, 298  
 Profit function, 296, 304  
 Promotion, 34, 58, 81, 95, 97, 98  
 Prompts, 32  
 Protection, 36  
 Pseudocost Branching (PB), 327  
 PwC, 15  
 Python, 116, 203  
 Python library, 116, 246  
 Pyton, 260  
 PyTorch eco-system, 232
- Q**  
 Quantile Regression Forest (QRF), 91–93  
 Quarterly, 195, 201, 206, 208, 213, 215, 271
- R**  
 Random, 67, 69, 92, 99, 132, 176, 194, 197, 205, 215, 379  
 Random Forest (RF), 86, 91, 92, 94, 96, 100, 225, 231, 236, 244, 245  
 Randomisation, 177  
 Random noise, 69  
 Ranking, 184, 223–225, 320, 356, 357  
 Raw data class, 232, 234  
 Real concept drift, 166  
 Real life applications, 8, 11, 53  
 Real Life Happenings/Events, 10, 12  
 Real-Time Recurrent Learning (RTRL), xxiv  
 Re-centering, 64  
 Recognition, 3, 9, 14, 18, 42, 88, 223  
 Recommendation Helpers, 12, 14  
 Rectified activation functions, 63  
 Rectified Linear Unit (ReLU), 55, 63  
 Recurrence plots (RPs), 259–261  
 Recurrent Neural Networks (RNNs), 62, 88, 97, 113, 116, 129, 168  
 Recurrent Self-Organizing Neural Fuzzy Inference Network (RSQFIN), xxiv

- Recurring Concept Drift (RCD), 166, 167, 169
- Reduced Error Pruning Tree (REPTree), xxxiv
- Regression Based Latent Factors (RLFs), xxxiv
- Regression techniques, 55, 56
- Reinforcement learning, 331
- Reinforcement learning (RL), 6
- Re-scaling, 64
- Researchers, 6, 23, 41, 50, 56, 60, 61, 94, 113, 116, 118, 170, 191, 193, 258, 314
- Residual errors, 53, 57, 61
- Residual Sum of Squares (RSS), 172–174
- Resources, 34, 36, 51, 63, 77, 203, 214, 215, 375, 380, 383
- Retail, 57, 78, 93, 95, 96, 100, 108, 225, 226, 229, 230, 232, 352
- Retail industry, 100, 222
- Rethink Robotics*, 15
- Retraining models, 215
- Ridged regression, 344
- Risk, 17, 35, 39, 40, 49, 77, 78, 269, 353, 365, 381, 383
- Roboticians, 15
- Robots, 6, 10, 15, 16, 19, 26, 34
- Robust Adaptive Dynamic Programming (RADP), xxxiv
- Robustness, 63, 89, 99, 100, 164, 195, 197, 204, 206, 215, 255, 265, 267, 381
- Robust optimisation, 321
- Robust SAA, 321
- Rodney Brooks, 15
- Root Mean Squared Error (RMSE), 177–179, 181, 182, 184–186, 362
- Root Mean Squared Propagation (RMSProp), 63
- Root Mean Squared Scaled Error (RMSSe), 245, 270
- Russia, 39
- S**
- Sales, 38, 78–80, 83–85, 89–93, 95–99, 101, 107, 108, 226, 230, 234, 240, 244, 258, 282
- Sales Review, 381
- Sample Average Approximation (SAA), 321, 322
- Samples, 169, 193, 197, 198, 200, 202, 203, 321, 328, 334, 341
- Saturation, 83, 85, 144
- SBoF model, 261
- Scaled Mean Absolute Error (scMAE), 199, 209, 211–213
- Scale invariance, 139
- Scenario, 4, 13, 18, 19, 23, 35, 36, 39, 40, 43, 84, 125, 133, 137, 147, 151, 213, 215, 228, 287, 305, 323
- Schwarz Information Criteria (BIC), 362
- Scrutable, 26
- Seasonal differences, 260
- Seasonality, 53, 94, 95, 97, 167, 228, 229, 257, 258, 260
- Security, 39, 40
- Selection of base forecasters, 228
- Self-driving, 8, 12, 13, 24, 333
- Self-learning, 55, 98
- Self-organizing map (SOM), 224
- Series weighted methods, 166, 171
- Set of time series, 111, 115, 126–128, 130, 132, 134, 136–139, 153, 168, 211, 222, 228, 229, 265
- Sigmoid curves, 144, 155
- Sigmoid model, 156–159
- Similar products, 78, 79, 81, 101
- Simple Exponential Smoothing (SES), 224, 236, 245

- Simulated multiseasonal time series, 257  
 Singapore, 8  
 Single step, 377  
 Siri, 9, 12  
 Skills, 20, 32, 34–36, 86  
 Smartphones, 80  
 Smart Predict and Optimise (SPO), 317  
 Smart Speakers, 12  
 Smooth Support Vector Machine (SSVM), xxxiv  
 Social Adaptive Ensemble (SAE), 165, 170  
 Social Network, 170  
 Societal expectations, 42  
 Society, 15, 17, 32, 33, 37, 41, 43  
 Software, 17, 21, 34, 40, 86, 99, 222, 324, 335, 377, 379, 381  
 Sophisticated, 23, 42, 52, 57, 65, 195, 264, 305  
 South Korea, 40  
 Sparsity, 117, 152, 280  
 Spatial information, 261  
 Spatial pyramid matching (SPM), 261  
 Specialties, 43  
 Spectrum, 16  
 Spectrum of the predict, 323  
 Speech, 3, 5–7, 9, 14, 21, 23, 88  
 Speech-to text, 11  
 S-shaped curves, 85, 90  
 Stairstep report, 376, 377, 379  
 Standard benchmarking datasets, 324  
 StarCraft, 9, 11, 18  
 State-of-the-art, 62, 113, 176, 194, 195, 203, 302, 315  
 State space models, 113  
 Static-global, 284  
 Stationary, 130, 163, 164, 168, 186, 256  
 Statistical modeling, 78  
 Statistical testing, 166, 179, 182, 183  
 Statistical trade-offs, 126, 135, 138  
 Stochastic, 89, 140, 193, 197, 198  
 Stochastic Gradient Boosting (SGBoost), xxxiv  
 Stochastic Gradient Descent (SGD), xxxiv  
 Stochastic programming, 321, 322  
 Stochastic value added (SVA), 383  
 Stochastic Variance Reduced Gradient (SVRG), xxxiv  
 Stock, 280–282, 284, 286, 287, 292, 297, 317, 324, 353  
 Stock Keeping Units (SKUs), 280  
 Strength, 38–40, 50, 54, 64, 78, 81, 88, 101, 137, 229, 325  
 Strong Branching (SB), 326  
 Structural components, 50, 54  
 Structured Nonlinear Parameter Optimization Method (SNPOM), xxxiv  
 Structures, 18, 34, 81, 89, 101, 235  
 Superintelligence, 15  
 Supervised, 97, 224, 226, 230, 231  
 Supervised classification, 327  
 Supervised Descriptor Learning (SDL), xxxiv  
 Supervised Linear Dimension Reduction (SLDR), xxxiv  
 Supervised ML, 315, 328, 330, 335  
 Supply, 26  
 Support inventory management, 91  
 Support Vector Machine (SVM), 244  
 Support Vector Ordinal Regression (SVOR), xxxiv  
 Support Vector Regression (SVR), 54  
 Symbolic Network Reliability Functions (SNRFs), xxxiv  
 Symmetric MAPE (sMAPE), 199, 208–210, 375
- T  
 Taiwan, 39

- Technological advances, 32  
 Technological singularity, 33, 35  
 Technology adoption, 81  
 Telecommunication, *xxi*  
 Temporal outbreak detection, 256  
 Tensors, 290–292  
 Tesla, 8, 12, 13  
 Text recognition, 5–7, 9, 14  
 Text-to-speech, 11  
 Text to voice transcriptions, 9  
 Theil’s-U, 375  
 The Mean Absolute Scaled Error (MASE), 199, 202  
 Theta, 109, 124  
 TikTok, 39  
 Time-adaptive Support Vector Machine (TA-SVM), *xxxiv*  
 Time Series, 123–147, 149–153, 155, 158–161, 163–168, 173, 176–179, 182, 186, 187, 191–195, 198–203, 210, 341, 359, 360  
 Time series, 78, 79, 82, 95, 221, 222, 224–226, 228–232, 234, 235, 238, 240, 244, 246, 247, 373, 377  
 Time series forecasting, 49, 50, 54, 56, 107, 108, 113, 114, 221, 222, 224–229, 232, 236, 246, 382  
 Time series generation, 69, 255, 256, 258  
 Time series imaging, 255, 259, 260, 268, 272  
 Time series interpolation, 69  
 Total percentage error (TPE), 382, 383  
 Trade, 38–40, 84  
 Traditional model classes, 134  
 Traditional performance measures, 375, 383  
 Training data, 92, 111, 114, 138, 163, 181, 202, 241, 253, 255, 315, 331  
 Training process, 55, 101, 139, 194, 195, 198, 203, 215, 305  
 Training set, 92, 178, 185, 187, 194, 299, 301, 302  
 Transfer information, 117, 161  
 Transfer-learning, 57, 68  
 Transformation, 52, 55, 59, 95, 125, 134, 135, 145, 153, 154  
 Transformer-based Forecasting, 301, 303, 306  
 Transformer networks, 88  
 Transform function, 234, 236  
 Transhumanism, 23  
 Transition function, 332, 334, 343, 344  
 Transparency, 33  
 Travelling Salesman Problem, 334  
 Tree-based models, 96, 119, 318, 334  
 Trend, 19, 26, 32, 84, 86, 90, 94–96, 115, 118, 126, 150, 161, 164, 167, 198, 228, 229, 379  
 Trimmed and winsorized averages, 66  
 Truly new-to-the-world products, 90, 101  
 Tsfeatures, 229, 260  
 Tsfmeta, 232, 236, 238, 244  
 Tsfresh, 230, 260
- U**
- Uber, 13  
 Ukraine, 39  
 Uncertainty, 8–10, 12, 85, 91, 192, 313, 315, 316, 320, 336, 383  
 Uncharted territory, 341  
 Uncorrelated stochastic data, 261  
 Underestimated, 40  
 United Kingdom, 155, 156  
 United States (US), 38, 113, 154, 350

- Univariate time series, 57, 62, 224  
Univariate time series forecasting, 214, 215  
University of California, San Francisco, 22  
Unpredictability, 10, 197  
Unsupervised, 89, 98, 224, 230, 239, 348, 358, 362
- W**  
Waymo, 12, 13, 17  
Weighted linear combination, 225  
weighted MAPE (wMAPE), 96, 375  
Welfare, 42  
Wholesale, 93, 100  
Window, 57, 110–112, 194, 200–202, 226, 234, 356
- V**  
Value added (VA), 52, 374, 377  
Variance, 33, 43, 52, 96, 137, 151, 165, 255, 264, 265  
Vector ARIMA, 56  
Virtual concept drift, 166  
Visibility, 95  
Visualization, 224
- X**  
XGBoost (XGB), 87, 96, 98, 99, 253, 270
- Z**  
Zalando, 280, 281, 288, 297, 306  
ZeroR, 92, 94