ARTICLE IN PRESS

International Journal of Forecasting xxx (xxxx) xxx

Contents lists available at ScienceDirect

International Journal of Forecasting

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

# Understanding machine learning-based forecasting methods: A decomposition framework and research opportunities

Casper Solheim Bojer

*Department of Materials and Production, Aalborg University, Fibigerstræde 16, 9220 Aalborg Øst, Denmark*

ARTICLE INFO

*Keywords:*
Machine learning
Forecasting
Ablation testing
M5 competition
Decomposition
Framework
Kaggle

ABSTRACT

Machine learning (ML) methods are gaining popularity in the forecasting field, as they have shown strong empirical performance in the recent M4 and M5 competitions, as well as in several Kaggle competitions. However, understanding why and how these methods work well for forecasting is still at a very early stage, partly due to their complexity. In this paper, I present a framework for regression-based ML that provides researchers with a common language and abstraction to aid in their study. To demonstrate the utility of the framework, I show how it can be used to map and compare ML methods used in the M5 Uncertainty competition. I then describe how the framework can be used together with ablation testing to systematically study their performance. Lastly, I use the framework to provide an overview of the solution space in regression-based ML forecasting, identifying areas for further research.



## 1. Introduction

The large progress made in the field of machine learning (ML) has started to spread to the field of forecasting spurring the development of new methods such as autoregressive neural networks (Benidis et al., 2020). Methods based on ML have already shown impressive performance in the M4 (Makridakis, Spiliotis, & Assimakopoulos, 2020) and M5 (M Open Forecasting Center, 2020) competitions, as well as multiple Kaggle competitions (Bojer & Meldgaard, 2021). ML methods based on neural networks and gradient boosting took the 1st (Smyl, 2020) and 2nd place (Montero-Manso, Athanasopoulos, Hyndman, & Talagala, 2020) in the M4 competition and presented substantial improvements over benchmarks in the modelling of forecast uncertainty (Makridakis et al., 2020). Neural networks and gradient boosting have also managed to dominate the four most recent of six forecasting competitions on the data science platform Kaggle (Bojer & Meldgaard, 2021). One of the main hypotheses for their strong performance is the use of cross-learning, where multiple time series are predicted using a single model, thus allowing for learning patterns across time series (Bojer & Meldgaard, 2021; Makridakis et al., 2020; Montero-Manso & Hyndman, 2021; Semenoglou, Spiliotis, Makridakis, & Assimakopoulos, 2020; Spiliotis, Makridakis, Semenoglou, & Assimakopoulos, 2020).

Despite the impressive empirical performance of ML methods, little is known about why, when, and how these methods outperform proven methods such as exponential smoothing. While the recent competitions highlighted the potential of ML methods, they also clearly demonstrated that not all ML methods are able to beat the statistical benchmarks methods (Makridakis et al., 2020). The same ML algorithm could significantly outperform the statistical benchmarks by a large margin in some solutions and fail to do so in others. Understanding why this is the case is a key research challenge for advancing the use of ML methods in forecasting.

The ML forecasting methods used successfully in recent forecasting competitions can be roughly divided into two classes, namely regression-based ML and neural forecasting methods. Regression-based ML transforms the

*E-mail address:* csb@mp.aau.dk.

https://doi.org/10.1016/j.ijforecast.2021.11.003

**Table 1**
Decomposition framework for regression based ML methods.

| Area | Component | Description |
|---|---|---|
| Preprocessing | Outliers | Handling unusually large or small values |
| | Imputation | Handling of missing values |
| Dataset construction | Feature engineering | Transforming time series into useful data representations |
| | Target engineering | Transforming the prediction target |
| | Training set construction | Strategies for weighting, subsetting, and augmenting the training data |
| Model training and validation | Model | Consists of an algorithm such as LightGBM and a loss function used to estimate parameters |
| | Multi-step strategy | Strategy used to generate multi-step forecasts |
| | Pooling/cross-learning | Strategy used to assign time series to one or more models |
| | Model evaluation | Evaluation strategies for model selection and performance evaluation |
| | Hyperparameter tuning | Strategy for exploring the space of model hyperparameters |
| | Final prediction strategy | Strategy for final predictions, e.g., re-training the model or averaging predictions of multiple historical models |
| Ensembling | Ensembling/combination | Methods for constructing and combining multiple models |
| Postprocessing | Postprocessing | Forecast adjustments |

time series prediction problem into a regression problem, whereas neural forecasting methods use architectures that enable directly processing time series and generating useful representations from them.

I argue that the main difficulty in understanding ML forecasting methods lies in their complexity. This is especially true for regression-based ML methods, as they often consist of many components in addition to the model, such as preprocessing, feature engineering, hyperparameter tuning, etc. Contestants in forecasting competitions have reported on their methods using different terminology and only on certain aspects of their solutions. Without a common language and abstractions for reporting on and studying these methods, it becomes difficult to identify the parts of the method design that result in performance differences.

In this paper, I attempt to contribute to the study of forecasting ML methods by:

- presenting a framework for regression-based ML forecasting methods that aims to provide a common abstraction and language for reporting on and studying them.
- demonstrating how the framework can be used to map complex ML methods using the winner and 3rd place of the M5 Uncertainty competition as examples.
- suggesting an approach for further study of regression-based ML methods by using the framework together with ablation testing.
- using the framework to identify underexplored areas ripe for further research.

## 2. Framework

Inspired by the ML process and building on a review and categorization of many ML solutions to forecasting problems in the M4, M5, and Kaggle competitions, I have developed the framework for ML forecasting methods as presented in Table 1. The framework has been shaped by the fact that most of the analysed solutions were of the regression-based ML class, and as such it is most suited for their analysis. It can be applied to map neural forecasting methods, although a decomposition specifically designed for this purpose should prove more insightful. In the framework, an ML method consists of components in five areas that are based on their purpose in the ML process:

- Preprocessing
- Dataset construction
- Model training and validation
- Postprocessing
- Ensembling

Each of these areas contains one or more components that I have grouped into coherent units. The framework is similar to existing models of the ML process (Amershi et al., 2019) and AutoML pipelines (He, Zhao, & Chu, 2021) in its organization. It differs from both by being focused on forecasting and thus includes multi-step strategy, pooling, and final prediction strategy as components. In addition, it differs from (Amershi et al., 2019) by being focused on decomposing the final method into its components rather than on representing the overall process. Compared to He et al. (2021), the framework includes ensembling and postprocessing but does not consider hyperparameter optimization for neural networks separately.

The following subsections will address each of the five areas and their components in detail and describe strategies observed in the competitions.

### 2.1. Preprocessing

The act of preprocessing and cleaning data is an important part of any quantitative forecasting method, although

it is often given little attention when the method is written up and published due to its' often context-specific nature. Two classes of cleaning come up often in time series applications: handling outliers and imputing missing values.

#### 2.1.1. Outliers

Outliers are unusually low or high values occurring due to data errors or special circumstances such as one-off events. Examples could be sales during a special one-off promotion event or webpage visits during a period of server downtimes. These unusual values can be problematic for traditional and ML methods for time series forecasting as they are both based on the extrapolation of historical patterns.

Unusually, large or small values can have a large impact on the forecast (Chen & Liu, 1993), which is undesirable if the outlier event is not expected to reoccur in the forecasted period. However, one should be careful about removing or changing outliers, as that might lead to underestimation of uncertainty when the event could potentially reoccur in future.

#### 2.1.2. Imputation

Imputation is concerned with replacing missing values in time series, as many time series methods do not handle missing values. Many methods exist for imputing missing values ranging from simple univariate methods such as last observation carried forward variations, interpolation, and model-based methods (Moritz & Bartz-Beielstein, 2017) to more complex methods such as matrix factorization (Yu, Rao, & Dhillon, 2016).

### 2.2. Dataset construction

Dataset construction is concerned with the process of transforming one or more time series into a suitable representation for an ML model. Training of ML models requires two inputs: (1) the input dataset **X** that the model is expected to learn patterns from and (2) the targets that the model will learn to predict **Y**. In the context of forecasting **Y** consists of future values of the time series, while **X** can consist of the raw historical time series data and other informative features, such as external factors or summaries of the historical data.

#### 2.2.1. Feature engineering

Feature engineering is concerned with transforming the time series dataset into the input dataset **X**. The goal of feature engineering is to come up with a set of transformations that increases the predictive performance of the model. The challenge here is not only coming up with potentially useful transformations, but also selecting which features to use, as including many irrelevant features are likely to lead to overfitting. Typical features include external factors, lagged values of the time series, rolling statistics, and other time series features (Bojer & Meldgaard, 2021). Feature selection is often done manually using cross-validation performance or in the case of the tree-based method using permutation importance.

#### 2.2.2. Target engineering

Target engineering is the process of making transformations to the target **Y** to improve predictive performance. Transformations to the target, or the time series itself, are typically made for one of the following three purposes: (1) making the time series stationary, (2) dealing with non-constant variance, and (3) scaling the time series. Stationarizing the time series is often done for traditional time series methods such as ARIMA, using differences or seasonal differences. The forecasting field, therefore, has a lot of tools already available for this area that should also prove useful for regression-based ML methods. The same is true for dealing with non-constant variance, where methods such as the Box–Cox transform (Box & Cox, 1964) are well-developed and understood.

The last issue of scaling has not been as relevant in the forecasting field until recently, as most modelling has focused on one model per time series. However, it becomes important when multiple time series are to be modelled using a single model, as patterns can otherwise not be learned across time series with different scales. Examples of scaling methods include variations of mean scaling where the time series is divided by its historical mean (Salinas, Flunkert, Gasthaus, & Januschowski, 2020) or a moving average.

#### 2.2.3. Training set construction

Training set construction concerns what data is used in model training and the importance assigned to each datapoint. It is often overlooked but is an important design parameter, as it together with feature and target engineering determine what is learnable from the data. Three key choices in the training set construction concerns subsetting, data augmentation, and weighting. Subsetting concerns not using a part of the data for model training. Examples would be to remove older data, as the patterns might have changed over time, or selective inclusion of data based on the period to be forecasted. As an illustrative application of the latter, some Kaggle contestants have removed holiday data for non-holiday forecasts or data from the winter period if a forecast was required for the summer period.

Augmentation is concerned with creating additional data by making changes to the original data. The challenge is in finding meaningful transformations that still preserve the patterns in the data, thus providing additional samples. An example of augmentation from the forecasting literature includes bagging of exponential smoothing models using an STL decomposition (Bergmeir, Hyndman, & Benítez, 2016). The winners of the M5 Uncertainty competition (ref – part of special issue) used augmentation in the form of randomly scaling the time series by a factor to improve extrapolation performance.

Weighting data can be used to assign different levels of importance to observations in terms of impact on model estimation. It can be used both to weigh errors in the loss function as part of the training process, which is supported by almost all modern ML methods, or as part of the training set construction to randomly resample from the dataset with probability in proportion to the weights.

### 2.3. Model training and validation

Model training and validation is the area that usually receives the most attention in research. It is concerned with all aspects related to the model, including its design, training, and validation. Two of the components are a result of issues specific to the forecasting context, namely multi-step strategy and final prediction strategy.

#### 2.3.1. Model

The model itself can be any supervised ML model that maps from the input data **X** to an output **Y.** The form of **Y** depends on the multi-step strategy and loss function. To optimize parameters in the training process, the model uses a loss function that assigns a cost to each prediction error. Common loss functions include mean squared error, which leads to a mean prediction, and quantile loss, which leads to a quantile prediction (Kolassa, 2020). An example use case for quantile loss is using multiple instances of the same model to forecast different quantiles, which allows for predicting the forecast distribution using standard regression-based ML methods.

#### 2.3.2. Multi-step strategy

One of the key challenges in using regression-based methods for time series is how to produce multi-step ahead forecasts. The work of Bontempi, Taieb, and Le Borgne (2012) presents four classes of strategies for multi-step forecasting: the recursive strategy, the direct strategy, the hybrid strategy *DirRec*, and multiple output strategies. In addition to the above strategies, Kaggle contestants have come up with three other strategies. The first, which I will call *seasonal direct*, uses one model per seasonal period to forecast ahead. In the M5 competition where four-week forecasts were required for daily data, this would mean using four models. This is possible by only including time series features that rely on seven-day (or more) lagged values. The second, which I will call *horizon modelling*, uses a single model to forecast multi-steps ahead without relying on a multi-output model. This is possible by including the horizon as a feature in the model and suitable constructing the training data. More concretely, each row in the input data matrix will be duplicated h times, a new column with the horizon will be added, and the corresponding h-step ahead values in Y will be aligned with the input matrix. The last strategy, which I will call *naïve regression* due to the discarding of recent information, is to simply ignore the multi-step aspect by only relying on lag-h or older values in the features used.

#### 2.3.3. Pooling/cross-learning

Pooling or cross-learning can be viewed as a spectrum of strategies for assigning time series to one or more models, ranging from local models with one model per time series to global models with a single model for all time series (Montero-Manso & Hyndman, 2021). While the use of local and global methods is straightforward, hybrid models are trickier as they also require a logic for assigning the time series to models, including deciding on the number of models to use. Examples of this logic include using the time series hierarchy, e.g., by having one model per store or product category or using time series clustering to decide on the subsets (Semenoglou et al., 2020).

#### 2.3.4. Model evaluation

Model evaluation is perhaps the most crucial component of an ML method, as it is responsible for preventing overfitting. Models can be evaluated for two separate purposes: model selection and performance evaluation. The purpose of model selection is to select the best "model", where the model in this context refers to the configuration of all the listed components. The selection is typically based on performance on one or more validation sets, potentially considering both the mean and variance of the forecast accuracy. The purpose of performance evaluation is to provide an unbiased assessment of the out-of-sample accuracy of the selected model. Strategies for model evaluation are well-known within the statistics and forecasting fields and include hold-out datasets, K-fold cross-validation, and time series cross-validation and variations such as blocked K-fold (Bergmeir & Benítez, 2012). A less known strategy used by some Kaggle contestants is a variant of K-fold cross-validation, known as grouped K-fold cross-validation in the ML community. This strategy uses a grouping variable and ensures that rows from the same group end up in the same fold. As an example of the strategy, the winners of the M5 Uncertainty challenge used year as the grouping variable, ensuring that data from the same year is not used for both training and evaluation.

#### 2.3.5. Hyperparameter tuning

Hyperparameter tuning is concerned with strategies for selecting the set of hyperparameters for a model that optimizes predictive performance. Hyperparameters typically specify how the model training process is carried out and tend to have a large impact on model accuracy and overfitting. The number of hyperparameters in ML methods makes it infeasible both from an overfitting and a computational perspective to conduct a brute-force search. Multiple search strategies have therefore been devised for finding good hyperparameters within a computational budget ranging from simple approaches, such as manual, grid, and random search, to genetic algorithms and Bayesian optimization (see Feurer & Hutter, 2019 for an overview)

#### 2.3.6. Final prediction strategy

The process of evaluating models using cross-validation or time series cross-validation results in multiple models being produced; one for each of the K-folds. After model selection, a choice must be made regarding how to generate out-of-sample forecasts for the test set. The typical approach would be to retrain the model using the newest data and use this retrained model. A risk with this approach is that the retrained model will not have had its performance evaluated and might perform poorly. Alternative approaches observed in Kaggle competitions include using the model from the most recent fold or using an average of the predictions from the K models. While these approaches do not use the most recent data, they are more robust as they have been evaluated thus reducing the risk of overfitting.

### 2.4. Ensembling/combinations

Ensembling, or combinations, concerns combining the output of multiple forecast models and very often leads to improved performance. Two main aspects are present with ensembling: how to create a diverse pool of forecasts models to combine and how to combine the forecasts. With time series models, diversity in the ensembles has typically been achieved through the inclusion of different model families or by combining different models from the same model family. With modern forecasting methods, diversity can also be added by modifying any of the components, e.g., features, hyperparameters, or data construction.

Approaches for combining the forecasts of the pool of models have been studied extensively in the forecasting community already (e.g., Kourentzes, Barrow, & Petropoulos, 2019). One of the two main approaches is stacking, where a model is trained to optimally combine the predictions from the pool of models. This introduces a separate model that can be designed using the components from the framework. The other approach is the use of simple combination operators such as the average or median.

### 2.5. Postprocessing

Postprocessing is concerned with adjusting the outputs generated by a model. Typically, this includes the use of the subjective expertise of the forecaster in the adjustment process. Examples include making trend adjustments to forecasts or clipping the forecasts, such as replacing negative forecasts with zero values or providing a max value for forecasts to ensure they stay within reasonable bounds. A widely researched data-driven approach suitable for hierarchical time series is forecast reconciliation, which exploits the structural information of the hierarchy and in addition to reconciliation often improves forecast performance at different aggregation levels (e.g., Wickramasuriya, Athanasopoulos, & Hyndman, 2019).

## 3. Application

In this section, I show the utility of the framework by demonstrating how it can be used to map and compare ML methods in the M5 Uncertainty competition. Due to space limitations, I use the winning (ref to be added – part of special issue) and the 3rd place solutions of the M5 Uncertainty competition as illustrative examples. A mapping of other reported solutions in the top 15 can be found in the supplementary materials. Note that the application is based on my interpretation of the published material from the contestants and thus slight inaccuracies are possible. Table 2 presents the mapping of the two solutions based on the framework.

Based on the mapping, it is evident that, despite relying on similar input data, the methods use different strategies in all the components, thus clearly showcasing the large design space and complexity in ML methods. Notably, both methods in Table 2 are similar in configuration to gradient boosting and neural network methods that have been successful in previous Kaggle competitions (Bojer & Meldgaard, 2021). Noticeable differences for the winning solution compared to the earlier Kaggle solutions include the use of reconciliation, horizon modelling, group K-fold, sampling, and range blending. The 3rd place solution differs mainly in terms of the use of oversampling and distribution parameters as model outputs. By making differences transparent, the framework aids in understanding the methods, but it does not by itself explain the performance differential. The large number of contestants using both LightGBM and neural networks with varying performance does, however, suggests that it does not lie in the model itself.

## 4. Understanding method performance

To truly learn from and understand the high performing methods in the M5 and other competitions, we need to understand which components are responsible for their performance. Ablation testing is a well-known method in the field of ML used to identify which parts of the often highly complex methods are responsible for improved performance and which merely add accidental and unnecessary complexity. It is standard to evaluate new neural network architectures by comparing the architectural innovations to benchmark modules or a reference architecture without the module. Ablation testing works by decomposing the method into components and then systematically varying or turning off specific components and measuring the impact on performance. In the case of the M5 Uncertainty winner, ablation testing the non-standard components in the solution seems like an obvious place to start, such as postprocessing, range-blending, weighted subsampling, and removal of holiday months.

Adopting ablation testing in forecasting research would have implications for our practices as a research community. In the field of ML, ablation testing is supported by a culture of using common benchmark datasets for testing, of open sourcing code, and of adopting modular software architectures for the methods developed. While the first two practices are already strong points of the forecasting community, the adoption of modular architectures is not yet widespread. The use of a modular architecture allows for easily swapping components, increasing the level of abstraction at which model developers and ablation testers can work. Great work is already being done in this area, e.g., in Python packages such as sktime (Löning, Bagnall, Ganesh, Kazakov, Lines et al., 2019) and GluonTS (Alexandrov et al., 2020), and I strongly recommend the community to join these efforts.

## 5. Research opportunities

As a side product of developing the framework, the process allowed me to take stock of the strategies developed for each of the components in the competitions examined. This resulted in a necessarily incomplete overview of the solution space for the design of regression-based ML methods for forecasting. This

**Table 2**
Mapping of winner and 3rd place solution in M5 Uncertainty competition.

| Component | Winner | Third place |
|---|---|---|
| Outliers | | |
| Imputation | | |
| Feature engineering | Exponentially weighted moving averages, rolling statistics based on sales and de-trended sales. Categorical and calendar features. Gaussian noise added to month. Feature selection using cross-validation performance | Sales, prices, and calendar variables from the last 28 days. Weekday and holiday features. Price trend. Average of time series for individual model |
| Target engineering | Range blending. | Power transform on aggregated time series |
| Training set construction | Excluded holiday months. Weight-based sampling. | Oversampled two most recent years by factor 2 and 4 |
| Model | LightGBM with quantile loss | Long short-term memory (LSTM) model with categorical embeddings, predicting distribution parameters. Negative log-likelihood of negative binomial for individual model and student-t for aggregate model |
| Multi-step strategy | Horizon modelling | Multiple output |
| Pooling/cross-learning | One model for each aggregation level and quantile | One model for store/item level. One model for aggregated series |
| Model evaluation | Nested group K-fold with year as group. Leave-one-group-out for hyperparameter tuning | 4-fold time series cross-validation with each validation fold having 8 weeks of data |
| Hyperparameter tuning | Random search | Early stopping for number of epochs. Others unspecified |
| Final prediction strategy | Weighted average of K models | Retraining for aggregated model. No retraining for individual model |
| Ensembling/combination | K-fold models with multiple subsamples used for each. Combination using average | Average of multiple seeds for hyperparameter tuning. No ensembling for final predictions |
| Postprocessing | Reconciliation of Levels 1–9 mean forecasts and adjusted quantiles based on this | |

overview in turn allowed for the identification of several areas of the solution space that the forecasting field has not yet studied in depth. One highly relevant area where research is already well underway in the area of cross-learning and pooling. Identifying when and where different pooling strategies outperform local models would advance the field considerably. In relation to this topic, the areas of feature and target engineering seem critical for getting the most value out of cross-learning and investigating the dependencies between these areas seem like a potentially fruitful research area.

Another area deserving of further research due to its importance for ML methods is cross-validation. While great research has already been conducted in this area (e.g., (Bergmeir & Benítez, 2012), the introduction of new cross-validation strategies and considerable advances in ML methods suggests that this area might be worth revisiting. The same is true for the area of multi-step strategies (Bontempi et al., 2012), where new strategies have been presented since the seminal paper on the topic.

The topics of sampling, augmentation, and weighting of data, which were present in the examined M5 solutions, also seem worthy of study. Proven statistical workhorses such as exponential smoothing are built on the insight that both recency and seasonality are important. However, attempts to incorporate this insight, such as by subsamling or weighting the data, has not been explored in the context of ML methods. Lastly, an effort to disentangle the role of context-specific pre- and postprocessing in method performance would be highly useful, as it can help determine the generalizability potential of the methods.

## 6. Conclusion

Inspired by research methods used in the field of ML, I presented a framework for the study of regression-based ML methods for forecasting. The goal of the paper is to provide a common language and abstraction for the study of ML methods for forecasting, as well as a vision for how the framework might be used together with ablation testing for studying their performance. While the framework is based on many ML method designs as reported in multiple forecasting competitions, it is but one scholar's initial attempt to decompose and tame the complexity of modern ML methods. As with all models, they improve significantly when confronted with more data, and it is my hope that the framework will be used, discussed, critiqued, and refined in the upcoming work on studying the methods from the M5 competition and evaluate the many new ML methods that are surely to be developed in the near future.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ijforecast.2021.11.003.

## 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. (2020). GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, *21*(116), 1–6.

Amershi, Saleema, et al. (2019). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., & Callot, L. (2020). Neural forecasting: Introduction and literature overview. arXiv preprint arXiv:2004.10240.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213.

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., & Meldgaard, J. P. (2021). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, *37*(2), 587–603.

Bontempi, G., Taieb, S. B., & Le Borgne, Y. A. (2012). Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School* (pp. 62–77). Berlin, Heidelberg: Springer.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *26*(2), 211–243.

Chen, C., & Liu, L. M. (1993). Forecasting time series with outliers. *Journal of Forecasting*, *12*(1), 13–35.

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning* (pp. 3–33). Cham: Springer.

He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, *212*, Article 106622.

Kolassa, S. (2020). Why the best point forecast depends on the error or accuracy measure. *International Journal of Forecasting*, *36*(1), 208–211.

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.

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. J. (2019). Sktime: A unified interface for machine learning with time series. arXiv preprint arXiv:1909.07872.

M Open Forecasting Center The M5 competition. (2020). https://mofc.unic.ac.cy/m5-competition.

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.

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.

Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*.

Moritz, S., & Bartz-Beielstein, T. (2017). imputeTS: time series missing value imputation in R. *R Journal*, *9*(1), 207.

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. (2020). Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*.

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.

Spiliotis, E., Makridakis, S., Semenoglou, A. A., & Assimakopoulos, V. (2020). Comparison of statistical and machine learning methods for daily SKU demand forecasting. *Operational Research*, 1–25.

Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. (2019). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, *114*(526), 804–819.

Yu, H. F., Rao, N., & Dhillon, I. S. (2016). Temporal regularized matrix factorization for high-dimensional time series prediction. In *NIPS* (pp. 847–855).