`MetaARIMA`: Automatic Configuration of ARIMA using Classifier Chains

Vitor Cerqueira[1,2] **Ricardo Inácio**[1,2] **Carlos Soares**[1,2,3]

[1]Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
[2]Laboratory for Artificial Intelligence and Computer Science (LIACC), Porto, Portugal
[3]Fraunhofer Portugal AICOS, Portugal

Abstract ARIMA models are widely used for time series forecasting, but selecting their configuration is challenging and typically relies on heuristic approaches based on goodness-of-fit measures. We propose `MetaARIMA`, a meta-learning approach that frames ARIMA configuration as an algorithm selection problem. By modelling top quantile configurations as a function of extracted time series features, our method efficiently and effectively identifies top-performing ARIMA model configurations. During inference, the selection is refined with successive halving and maximal marginal relevance. Experiments on the M3 Monthly and Quarterly datasets, which comprise a total of 2184 univariate time series, show that `MetaARIMA` outperforms the standard `AutoARIMA` approach in terms of average rank.

1 Introduction

ARIMA (Auto-Regressive Integrated Moving-Average) models are widely used in time series forecasting (Box et al., 2015). While their effectiveness is well-established, selecting an adequate configuration is challenging. State-of-the-art quantitative approaches, such as `AutoARIMA` (Hyndman and Khandakar, 2008), rely on heuristics based on in-sample goodness-of-fit measures. Manual approaches such as the Box-Jenkins method (Makridakis and Hibon, 1997) require significant technical expertise which is difficult to scale to datasets composed of hundreds of time series or more. Further, these methods are usually memoryless in the sense that they do not take into account the performance of a given configuration on previous time series experiments.

We frame the problem of configuring ARIMA models as an instance of the algorithm selection problem (ASP) using meta-learning (Brazdil et al., 2008). The goal is to learn a meta-learning model (i.e., metamodel) that can select the best model configuration, or subset of configurations, for a given time series, based on the performance from previous experiments using other time series.

Predicting the single best performing configuration is challenging, given that finding the absolute best might be a matter of luck. Moreover, many ARIMA configurations may perform comparably well for a given time series. Therefore, instead of focusing on modelling the best configuration, we aim to predict which ones fall within a top quantile of performance. This frames the problem as a multi-label binary classification task, where multiple configurations can be considered among the top performers. We employ a classifier chain approach (Read et al., 2011), a state-of-the-art method for multi-label problems, using as input features extracted from each series.

During inference, rather than selecting only the most likely best configuration, we select a subset of k promising candidates. These are selected based on two criteria: i) predicted probabilities, which quantify the likelihood of a given configuration performing among the best ones; ii) maximal marginal relevance (MMR) (Cerqueira et al., 2019), to make sure that the subset of configurations is diverse. This approach acknowledges that there is a significant correlation among different configurations and includes an element that fosters diversity in the subset of configurations.

Among these, the final configuration is selected by minimizing the corrected Akaike Information Criterion (AICc) (Sugiura, 1978), a selection criterion that balances goodness-of-fit with model complexity and is well-established in the time series literature. We also leverage successive halving to speed up computations.

We evaluate our approach, dubbed as `MetaARIMA`, on the M3 benchmark dataset (Makridakis and Hibon, 2000). The results show that `MetaARIMA` consistently outperforms `AutoARIMA`, the state-of-the-art method for configuring ARIMA models, and other relevant baselines.

2 Problem Definition

Time series are sequences of data observed across time with a predefined sampling frequency. Formally, a time series Y of length t is defined as $\{y_1, y_2, \ldots, y_t\}$, where $y_i \in \mathbb{R}$ is the i-th value of Y. Forecasting denotes the process of predicting the next h values of Y, $y_{t+1}, \ldots, y_{t+h}$, based on historical data, where h is the forecasting horizon. Forecasting problems often involve time series collections that contain multiple univariate time series. We define a time series collection as $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_N\}$, where N is the number of time series in the collection.

Several methods have been developed to address forecasting problems. One widely used approach is ARIMA, a well-established linear model that combines three components: auto-regression, moving-average modelling, and differencing (Box et al., 2015). We provide a more in-depth description of this technique in supplementary material.

3 `MetaARIMA`

This section describes `MetaARIMA`, a novel approach for automatically configuring ARIMA models. `MetaARIMA` follows a meta-learning approach based on a classifier chain that models the top quantile configurations based on time series features. The methodology behind `MetaARIMA` is split into two parts. First, the process of collecting the metadata and building the meta-learning model is detailed in Section 3.1. Then, Section 3.2 details the inference step, where the metamodel is used to select a configuration for a new input time series.

3.1 Development stage

Figure 1 illustrates the workflow for building a meta-learning classifier chain. This process is split into three parts: i) meta-feature extraction; ii) performance estimation and correlation analysis; and iii) building the classifier chain model.

3.1.1 Performance estimation. Let $\mathcal{A} = \{A_1, \ldots, A_a\}$ denote the set of ARIMA configurations. We conduct a performance estimation procedure to collect data for meta-learning, specifically the meta-learning targets. First, each time series Y is split into Y_{train} and Y_{test} sets, where the latter is composed of the last h observations in each series, and h is the forecasting horizon. Then, for every

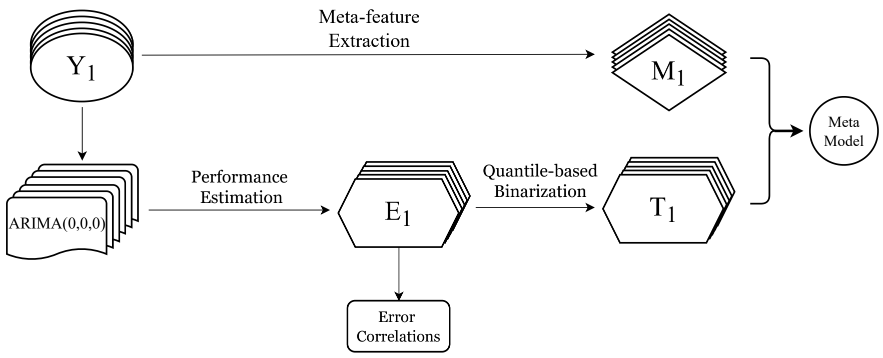

Figure 1: Workflow for building a meta-learning chain classifier.

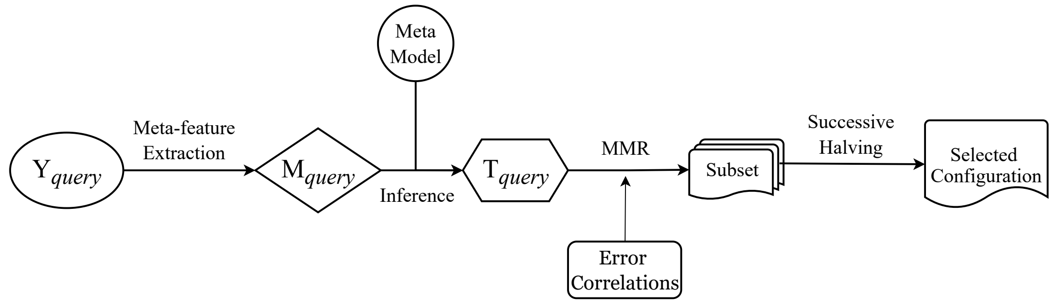

Figure 2: Workflow for the inference stage of `MetaARIMA`.

ARIMA model in the configuration pool, we fit it to each Y_{train} available. Afterwards, we evaluate each ARIMA model on the Y_{test} sets to estimate forecasting accuracy.

This process results in a set of error estimates for each time series $\mathcal{E} = \{E_1, E_2, \ldots, E_N\}$, where a given E_i denotes the performance scores of each configuration on Y_i: $E_i = \{e_i^1, \ldots, e_i^a\}$, where e_i^j is the score of the j-th configuration on the i-th time series. We assume the performance score is an error metric (e.g. SMAPE) that should be minimized.

We conduct a complementary correlation analysis using the errors scores $\mathcal{E}$. We compute Pearson's correlation to estimate the redundancy of each pair of configurations in $\mathcal{A}$. Let $R \in \mathbb{R}^{a \times a}$ denote the correlation matrix concerning the configurations $\mathcal{A}$, where R_{ij} is the correlation between configurations A_i and A_j, where $A_i, A_j \in \mathcal{A}$.

3.1.2 Meta-feature extraction.

For the meta-learning explanatory variables, we leverage time series feature extraction. Since different time series may vary in terms of scale and dimension, feature extraction is useful to bring them into a common representation. Each time series $Y \in \mathcal{Y}$ is transformed into a l-dimensional vector using some extraction technique. This process results in a set of meta-features $\mathcal{M} = \{M_1, \ldots, M_N\}$, where M_i is the set of l meta-features for time series Y_i.

3.1.3 Classifier chain.

We aim to conduct meta-learning using $\mathcal{M}$ and $\mathcal{E}$. The performance estimates $\mathcal{E}$ are discretized into binary values according to whether the corresponding configuration is on the top quantile of scores. Let τ denote a threshold value that determines the top quantile of configurations. For each time series Y_i, we compute a binary vector $T_i = \{t_i^1, \ldots, t_i^a\}$ where:

$$t_i^j = \begin{cases} 1 & \text{if } e_i^j \leq \tau_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where τ_i is the τ-th quantile of error scores for time series Y_i. This results in a set of binary vectors $\mathcal{T} = \{T_1, \ldots, T_N\}$, where T_i indicates which configurations achieved top performance for series Y_i.

The metadata for meta-learning is composed of input-output pairs $(\mathcal{M}, \mathcal{T})$, where $\mathcal{M}$ represents the meta-features and $\mathcal{T}$ represents the binary target variables indicating top-performing configurations. This denotes a multi-label classification problem, where each time series can have multiple top-performing configurations. We develop a classifier chain for meta-learning, which essentially turns a multi-label classification problem, into a linked sequence of binary problems, where each classifier adds to its input the predictions from its predecessors (Read et al., 2011; Pinto et al., 2016). This approach is useful to model the inter-dependencies between configurations.

3.2 Inference stage

Figure 2 illustrates inference process of `MetaARIMA` for a new query time series Y_{query}.

This stage starts by transforming Y_{query} into a set of meta-features M_{query}. We use these to get probabilistic estimates from the metamodel: $\hat{T}_{query}$. These probabilities denote $\mathbb{P}(e^j_{query} < \tau)$ for each configuration $A_j \in \mathcal{A}$.

3.2.1 Top-K selection.

Our approach for selecting the best configuration $A^* \in \mathcal{A}$ is to first select a top k based on $\hat{T}_{query}$ and the correlation matrix R, to then systematically evaluate and select one.

We construct a top k of configurations by ordering them by predicted probability $\hat{T}_{query}$, with the best predicted configuration being ranked first (i.e., argmax($\hat{T}_{query}$)). Then, we iteratively prune this subset using MMR. We iteratively remove a configuration if it is highly correlated with an already selected one. This results in a subset of k configurations that is diverse and predicted to perform well relatively to the overall pool of configurations.

3.2.2 Successive Halving and AICc-based final selection.

Instead of fitting all selected configurations on the complete training data, we also employ *successive halving* to accelerate the selection process. In this approach, we initially fit all candidate ARIMA configurations on a small subset of the training data. After evaluating their performance using AICc, only the top-performing half of the configurations are retained for the next round, where they are trained on a larger portion of the data. This process is repeated, progressively increasing the amount of training data and halving the number of candidates at each stage, until a single configuration remains.

4 Experiments

We evaluate the performance of `MetaARIMA` and compare it with state-of-the-art approaches and relevant baselines. We use the M3 dataset, specifically the monthly and quarterly subsets, comprising a total of 2184 time series. In terms of parameter setup, we set k to 20 and the quantile threshold to `0.1`. We use the default search process of `AutoARIMA`, available on the `statsforecast` Python package. We refer the reader to additional details in the supplementary material.

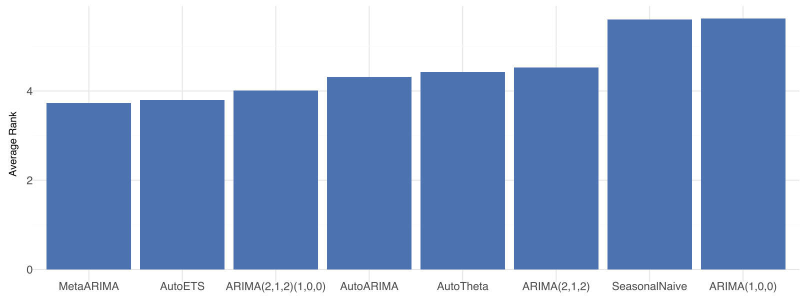

Figure 3: Average rank of each approach across all time series.

Figure 3 shows the average rank of each approach across all time series. `MetaARIMA` shows the best average rank, outperforming `AutoARIMA`, other fixed ARIMA configurations (`ARIMA(2,1,2)` and `ARIMA(1,0,0)`), and seasonal naive.

5 Conclusions

We present `MetaARIMA`, a novel approach for automatically configuring ARIMA models for univariate time series forecasting. `MetaARIMA` frames the task of configuring ARIMA models as a meta-learning-based ASP. It addresses this problem using a classifier chain approach that models top-performing configurations based on features extracted from time series data. During inference, successive halving is coupled with AICc to select the best configuration. Results in the M3 dataset (i.e., monthly and quarterly frequency subsets), comprising a total of 2184 time series, indicate that `MetaARIMA` outperforms `AutoARIMA` and other benchmarks effectively, at a lower computational cost.

Acknowledgements. This work was partially funded by projects AISym4Med (101095387) supported by Horizon Europe Cluster 1: Health, ConnectedHealth (n.º 46858), supported by Competitiveness and Internationalisation Operational Programme (POCI) and Lisbon Regional Operational Programme (LISBOA 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) and Agenda "Center for Responsible AI", nr. C645008882-00000055, investment project nr. 62, financed by the Recovery and Resilience Plan (PRR) and by European Union - NextGeneration EU, and also by FCT plurianual funding for 2020-2023 of LIACC (UIDB/00027/2020 UIDP/00027/2020);

References

Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric reviews*, 29(5-6):594–621.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer.

Anderson, O. (1977). The box-jenkins approach to time series analysis. *RAIRO-Operations Research*, 11(1):3–29.

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., and Gamboa, H. (2020). Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Brazdil, P., Carrier, C. G., Soares, C., and Vilalta, R. (2008). *Metalearning: Applications to data mining*. Springer science & business media.

Cerqueira, V., Torgo, L., Pinto, F., and Soares, C. (2019). Arbitrage of forecasting experts. *Machine Learning*, 108:913–944.

Chatfield, C. (2000). *Time-series forecasting*. Chapman and Hall/CRC.

Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77.

Dubey, A. K., Kumar, A., García-Díaz, V., Sharma, A. K., and Kanhaiya, K. (2021). Study and analysis of sarima and lstm in forecasting time series data. *Sustainable Energy Technologies and Assessments*, 47:101474.

Fulcher, B. D. (2018). Feature-based time-series analysis. In *Feature engineering for machine learning and data analytics*, pages 87–116. CRC press.

Hyndman, R., Kang, Y., Montero-Manso, P., O'Hara-Wild, M., Talagala, T., Wang, E., and Yang, Y. (2024). *tsfeatures: Time Series Feature Extraction*. R package version 1.1.1.9000, https://github.com/robjhyndman/tsfeatures.

Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27:1–22.

Makridakis, S. and Hibon, M. (1997). Arma models and the box–jenkins methodology. *Journal of forecasting*, 16(3):147–163.

Makridakis, S. and Hibon, M. (2000). The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476.

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., and Talagala, T. S. (2020). Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92.

Pinto, F., Soares, C., and Mendes-Moreira, J. (2016). Chade: Metalearning with classifier chains for dynamic combination of classifiers. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 410–425. Springer.

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2011). Classifier chains for multi-label classification. *Machine learning*, 85:333–359.

Rice, J. R. (1976). The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier.

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):1–25.

Sugiura, N. (1978). Further analysis of the data by akaike's information criterion and the finite corrections: Further analysis of the data by akaike's. *Communications in Statistics-theory and Methods*, 7(1):13–26.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4):437–450.

Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175.

A ARIMA

A.1 AR, I, and MA

The Auto-Regressive (AR) and Moving-Average (MA) approaches are two cornerstone techniques for modelling time series. According to the AR(p) model, the value of a given time series, y_i, can be estimated using a linear combination of its past p observations, along with an error term ϵ_i and a constant term c (Box et al., 2015):

$$y_i = c + \sum_{j=1}^{p} \phi_j y_{i-j} + \epsilon_i \tag{2}$$

where $\phi_j, \forall\, j \in \{1, \ldots, p\}$ are the model parameters, and p represents the order of the model, also often referred to as the number of lags.

Essentially, the AR(p) model uses the past values of the time series as explanatory variables in a multiple regression technique. Similarly, the MA(q) model uses past errors as explanatory variables:

$$y_i = \mu + \sum_{j=1}^{q} \theta_j \epsilon_{j-i} + \epsilon_i \tag{3}$$

where μ denotes the mean of the observations, $\theta_j, \forall\, j \in \{1, \ldots, q\}$ are the parameters of the model, and q denotes the order of the model. Hence, it essentially models time series according to random errors that occurred in the past q lags (Chatfield, 2000).

The ARMA(p,q) model combines AR(p) with MA(q):

$$y_i = c + \sum_{j=1}^{p} \phi_j y_{i-j} + \sum_{j=1}^{q} \theta_j \epsilon_{i-j} + \epsilon_i \tag{4}$$

and works under the assumption that a time series can be modelled as a linear combination of its own lags and past errors. However, it also assumes that the time series is stationary. This condition is too strict for many real-world phenomena showing a non-stationary structure, such as trends or seasonality. However, many interesting phenomena in the real-world exhibit a non-stationary structure (e.g., time series with trend and seasonality). ARIMA addresses this limitation by extending ARMA with an integrated (I) component of order d, leading to ARIMA(p,d,q). ARIMA stabilizes the level of a time series via successive differencing until it becomes stationary.

A.2 Seasonal ARIMA

When dealing with time series with a strong seasonal component, ARIMA is extended to include seasonal components. This process results in a SARIMA model (i.e., Seasonal ARIMA), which can be denoted as ARIMA(p,d,q)(P,D,Q)$_m$, where p,d,q are the non-seasonal parameters as described above. P,D,Q are their seasonal counterparts that apply the same AR, I, and MA components but at the seasonal level, and m is the number of observations per seasonal cycle (e.g., $m = 12$ for monthly data with yearly seasonality) (Dubey et al., 2021). The non-seasonal and seasonal components are combined linearly. The non-seasonal components (p, d, q) capture shorter-term dynamics, while the seasonal components (P, D, Q) model repeating patterns by applying the same AR, I, and MA operations at seasonal lags.

A.3 Configuring an ARIMA Model

Different configurations of ARIMA can be specified by defining the different orders of its components (p, d, q, P, D, Q). Selecting a configuration is a challenging problem and different frameworks are available in the literature to solve it.

Cross-Validation. A reliable out-of-sample approach for selecting an appropriate configuration. However, since we generally need to test an exhaustive amount of alternatives for each configuration, it becomes impractical at larger scales. For instance, if the number of considered configurations C is large, and each one has to be successively applied to refit and evaluate a model for a given number of F folds (Raschka, 2018), then it essentially leads to $C \times F$ fittings.

The Box–Jenkins technique. It is an iterative approach where different steps are applied to define an appropriate model based on the intrinsic autocorrelation of the data points (Box et al., 2015). The process starts by screening for stationarity, applying transformations to ensure that this property holds. Then, several parameters are probed to establish appropriate orders for the model components. This can be achieved by calculating and analysing the number of significant lags in ACF and PACF plots (Anderson, 1977), which are indicative of the MA (q) and AR (p) orders, respectively. After a plausible model configuration is identified (p,d,q), the model coefficients can then be estimated (Anderson, 1977), for example, by diminishing the overall magnitude of errors via nonlinear optimization (Zhang, 2003). The process ends when a diagnostic step indicates that the estimated model adequately fits the underlying data and assumptions about residuals. If that is not the case, new parameters must then be estimated. Although effective, this method requires expertise to be fruitfully implemented. Traditionally, analysts would apply domain knowledge for order and structure selection, which becomes unfeasible at larger scales.

The AutoARIMA methodology. This technique aims to solve the aforementioned problem by employing an automatic selection process to find the optimal ARIMA parameters for a given problem (Hyndman and Khandakar, 2008). This way, it is possible to obtain an effective model without the need of domain expertise (e.g., to inspect PACF or ACF plots) or an understanding of the data generating process. It first conducts some tests (e.g., KPSS) to find the differencing orders, then it leverages a given optimization technique, such as AICc (Akaike, 1998) and cross validation, to search for the best parameters for the problem at hand. However, this approach relies on goodness-of-fit to guide the optimization, which may lead to unreliable indications of predictive performance on unseen data (Tashman, 2000), potentially leading the model to overfit. Moreover, since this process heavily relies on heuristics, if the initial assumptions and carried search path are not the best, then the final model might just be a local optimum of all possible configurations.

Yet, besides the problems highlighted above, all of these approaches also have the limitation of being memoryless, which entails that the performance of different configurations on previous experiments is not exploited, which leads to large overheads and cost.

B Time Series Feature Extraction

Feature extraction provides a systematic way of describing time series. Different series may have a varied and potentially large number of observations. Thus, summarizing them into a common representation enables dimensionality reduction and more meaningful comparisons. Another benefit of using time series features is interpretability. Measures such as seasonal strength reveal interpretable insights from time series that are not easy to get from raw data points.

We formalize a feature extraction function g taking the form $\{Z_i^j\}_{j=1}^{l} = g(Y_i)$, where Z_i is the set of l features extracted from the time series Y_i. The set Z_i contains l different statistical and information-theoretic measures from a given time series. Different types of features are usually extracted, such as time-independent measures (e.g., mean, variance, or distribution entropy); measures of stationarity; autocorrelation values; frequency-domain measures; or nonlinear properties (Fulcher, 2018).

Several frameworks are available for extracting features from time series automatically. These include `tsfeatures` (Hyndman et al., 2024), `tsfel` (Barandas et al., 2020), or `tsfresh` (Christ et al., 2018). In particular, `tsfeatures` is often used in forecasting problems such as weighted forecast combination (Montero-Manso et al., 2020). This framework extracts a total of 42 features, such as linearity, lumpiness, or seasonal strength.

C Meta-learning for Algorithm Selection

C.1 Algorithm Selection Problem

For a given forecasting problem, a broad selection of algorithms is available for users to choose from, which fosters a reliable forecasting operation. However, the specificities of the models, of the data, and of the preprocessing tactics applied, are all crucial in determining forecasting performance in specific problem instances (Ahmed et al., 2010). Thus, identifying which model is optimal, or simply an effective approach for a given problem, can be challenging. The foundational work of (Rice, 1976), introduced the algorithm selection problem (ASP), which provides a systematic solution to such a challenge. It takes into account performance across different instances to objectively assess and identify prospective optimal algorithms. It highlights how features extracted from problem instances can be employed to ease the selection task, as these are commonly lower dimensional, thus simpler than raw values, while still being closely related to algorithm performance.

Hence, a conceptual methodology can be formulated for selecting the best algorithm. It starts by identifying instances in problem space, ($x \in \mathcal{P}$), from which features should be extracted ($f(x) \in \mathcal{F} = \mathcal{R}^m$), and mapped to feature space $\mathcal{F}$. Afterwards, selection mapping specifies which algorithm A from algorithm space $\mathcal{A}$ should be selected according to the extracted features ($S(f(x))$).

Performance mapping subsequently relates algorithm performance to the problem instance in performance measure space ($p(A, x) \in \mathcal{R}^n$), and the algorithm that maximizes this mapping is picked. It should be noted that selection relies on the features of x, but performance assessment still depends on both the algorithm and the instance (A, x), and since the first is usually lower dimensional ($m < n$), the task is made less cumbersome. Simply put, the goal is to maximize performance across algorithms per instance.

C.2 Meta-learning for Time Series

Based on the idea of using information about the performance of learning algorithms and data (i.e., meta-features) to learn a model, the concept of meta-learning was established. Although several lines of research emerged from it, such as algorithm ranking models, combination of algorithms, or self-adjusting algorithms (Smith-Miles, 2009), we focus on building a model that learns about the performance of learning algorithms and relates it to data features for the problem at hand, to effectively select an appropriate algorithm configuration.

D Data

We use the M3 (Makridakis and Hibon, 2000) dataset, specifically the monthly and quarterly frequency subsets. These are summarised in Table 1.

Table 1: Summary of the dataset: average value, number of time series, number of observations, seasonal period, and forecasting horizon.

Dataset	Average value	# time series	# observations	Period	h
M3 Monthly	117.3	1428	167562	12	12
M3 Quarterly	48.9	756	37004	4	8