TSPredIT: Integrated Tuning of Data Preprocessing and Time Series Prediction Models

Rebecca Salles[1], Esther Pacitti[2,3], Eduardo Bezerra[1], Celso Marques[1], Carla Pacheco[1], Carla Oliveira[1,3,4], Fabio Porto[4], and Eduardo Ogasawara[1](✉)

[1] Federal Center for Technological Education of Rio de Janeiro (CEFET/RJ), Rio de Janeiro, Brazil
{rebecca.salles,carla.pacheco}@eic.cefet-rj.br, {ebezerra,celso.silva}@cefet-rj.br, carla.oliveira@ibge.gov.br, eogasawara@ieee.org

[2] University of Montpellier, Montpellier, France
Esther.Pacitti@lirmm.fr

[3] National Institute for Research in Digital Science and Technology (INRIA), University of Montpellier, Montpellier, France

[4] National Laboratory for Scientific Computing (LNCC), Petropolis, Brazil
fporto@lncc.br

Abstract. Prediction is one of the most important activities while working with time series. There are many alternative ways to model the time series. Finding the right one is challenging to model them. Most data-centric models (either statistical or machine learning) have hyperparameters to tune. Setting them right is mandatory for good predictions. It is even more complex since time series prediction also demands choosing a data preprocessing that complies with the chosen model. Many time series frameworks, such as Scikit Learning, have features to build models and tune their hyperparameters. However, only some works address tuning data preprocessing hyperparameters and model building. TSPredIT addresses this issue in this scope by providing a framework that seamlessly integrates data preprocessing activities with models' hyperparameters. TSPredIT is made available as an R-package, which provides functions for defining and conducting time series prediction, including data pre(post)processing, decomposition, hyperparameter optimization, modeling, prediction, and accuracy assessment. Besides, TSPredIT is also extensible, which significantly expands the framework's applicability, especially with other languages such as Python.

Keywords: time series · prediction · data preprocessing · machine learning · hyperparameter optimization

1 Introduction

The prediction of time series has gained more attention in the last decades. Many time series prediction methods have been developed and can be found

A. Hameurlain et al. (Eds.): *TLDKS LIV*, LNCS 14160, pp. 41–55, 2023.
https://doi.org/10.1007/978-3-662-68014-8_2

in the literature [4]. An adequate prediction method is mandatory for building the right model [31], especially for data-driven models. They are generally organized between statistical and machine learning [20]. These types of methods usually have to set hyperparameters. In this sense, hyperparameter optimization is a fundamental step since it can influence the predictive performance of the resulting models [16,19].

Additionally, these models might be improved by adequate data preprocessing activities. Most of these methods tend to be optimistic regarding their assumptions over the time series and are not ready to handle nonstationarity [4,30]. They also suffer from the presence of concept drift [21] or lack of data [33]. These two cases are related. When concept drift occurs, generally, there are few samples to support model building. Usually, nonstationarity demands transformation methods to address this issue [30]. Besides, while working with small samples, data augmentation techniques are also needed [23,33,35].

For the algorithm to make predictions with greater accuracy, optimizing the hyperparameters is necessary [34]. Hyperparameters are values that make up the initial configuration of the learning algorithm [9]. Hyperparameters are also present in data preprocessing methods. Several factors influence the predictive performance of time series models, mainly choosing and tuning the right methods for data preprocessing and hyperparameters.

Regarding statistical learning, some methods seamlessly integrate hyperparameters optimization of data processing techniques. It includes the autoregressive integrated moving average (`ARIMA`) algorithm that optimize parameters (p, d, q) for `ARIMA` [12]. The d parameter represents the Integrated part of ARIMA and performs the differentiation of observations internally as a preprocessing step for the series to be stationary. The p parameter corresponds to the AR part of ARIMA, the number of autoregressive terms. The MA model works with the size of the moving average window and is represented by the q parameter. This method tunes altogether differentiation, autoregressive, and moving average models [3]. Conversely, there are many frameworks for machine learning, such as Scikit learn [11,26], which provides (i) a broad range of prediction methods, (ii) an extensive set of preprocessing methods, (iii) hyperparameter optimization features for machine learning. However, directly optimizing data processing and machine learning is left for users to program according to their needs.

In this context, this paper presents TSPredIT, an evolved version of TSPred [31] that seamlessly integrates the tuning of data preprocessing and time series prediction models for univariate time series. It only concerns regression models and is specialized integrating data transformation methods and data augmentation to aid in building machine learning methods (`MLM`) prediction models. TSPredIT is made available as an R-package. It is the first tool to seamlessly integrate a broad range of data transformation and preprocessing methods and state-of-the-art statistical and machine learning prediction methods for addressing nonstationary time series. The package automates the time series prediction process and parameterization while enabling user-defined prediction methods and data transformations, including code built in other languages like Python. Due to that, the features provided by TSPredIT are shown to be competitive regarding time series prediction accuracy.

Besides this introduction, the paper is organized into five more sections. Section 2 presents the background, while Sect. 3 presents the related work. Section 4 presents the TSPredIT, which evolves from the previous version of TSPred [31]. Section 5 provides a clear example of using TSPredIT's main features. Conversely, Sect. 6 characterizes the effect of choosing data preprocessing and different `MLM` during prediction. Finally, Sect. 7 concludes the work.

2 Background

Time series prediction is commonly associated with the scenario of regression. For simplicity, the paper may refer to prediction and regression interchangeably. Relevant models adopted for time series prediction generally fall into the categories of statistical or machine learning models [29]. The accuracy of the predictions depends on the quality of the historical data, the appropriateness of the model, and the assumptions made about the underlying processes driving the time series [10,14].

Figure 1 presents a general time series prediction process. It encompasses five main activities. It provides a general framework for predicting a time series based on a particular setup of preprocessing methods and prediction models. They are briefly described here, and some parts are detailed in the following sections.

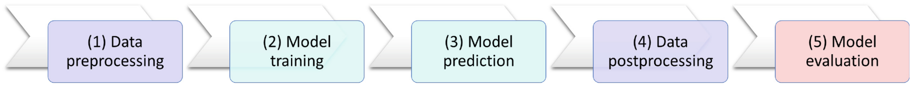

Fig. 1. Time series prediction process [29]

Activity 1, depicted in Fig. 1 in purple, refers to acquiring the time series and performing data preprocessing. It is generally associated with data cleaning, normalization, and transformation but might include other techniques, such as data augmentation. The transformations commonly change the time series domain values, and their parameters must be stored to support later detransformation to the original domain. For time series prediction, splitting the time series into a training and test set is also important during data preprocessing. All data preprocessing parameters should be computed during training and reapplied from the tune-values of training during the test. The model is built using the training slice and evaluated using the unseen test set, always ahead. However, when the goal is to adjust a model for the time series, the model does not need to be partitioned into a training and test set.

Activity 2, in blue, addresses model training. The prediction methods very often require hyperparameter optimization. In such a case, the training slice is again split into a novel training and validation set. Alternative models exploring hyperparameters values are built using the novel training and evaluated using the validation set. Once hyperparameters are fixed, a single model is built using the entire training dataset. From this moment, the model is available for use.

Activity 3, also in blue, refers to the model prediction. It is worth mentioning that the predicted values are not in the time series domain. In this sense, they can not be directly evaluated. Data postprocessing is needed in this case. Activity 4, also in purple, corresponds to the postprocessing of predictions, reversing transformations applied to the time series data in Activity 1. In a macro view, the data is normalized (scaled) and given as input to an algorithm. After the forecast, a denormalization process maps back the predicted values into the original scale of the time series. An example of postprocessing can be seen in previous work [24], where the data is denormalized to measure the error in the same scale for comparison purposes.

Finally, Activity 5, in pink, is the evaluation of prediction errors yielded by the model, as well as model fitness metrics. If the results are inadequate, this process can be revised and repeated to refine models. This process iteratively improves the quality of predictions (for time series prediction) or model adjustment (for time series modeling). The prediction can be evaluated in several ways, mostly measuring the errors between prediction and actual observation, such as Mean Square Error (`MSE`) and symmetric MAPE (`sMAPE`) in a test set. Alternatively, they can be measured by the level of model adjustment, such as Akaike Information Criterion (`AIC`) and Bayesian Information Criterion (`BIC`) [31].

This process provides a systematic way of predicting a time series based on particular preprocessing and prediction methods. It also focuses on prediction and model evaluation, that is, evaluating the accuracy of prediction and the fitness of a model. Such evaluation may indicate a demand for refining and perfecting the preprocessing-modeling setup and its parameters to obtain a more accurate model. This process may be repeated if the evaluated time series prediction model does not reach the desired accuracy. This process enables benchmarking different preprocessing-modeling setups.

3 Related Work

Several authors focused on the task of exploring different models. Ramey [27] and Lessmann et al. [18] developed frameworks for evaluating classification models and algorithms. Moreover, Bischl et al. [2] and Eugster and Leisch [8] developed the R-packages *mlr* and *benchmark*, respectively, which provide tools for executing automated experiments when benchmarking a set of models for data mining tasks such as classification and regression. These packages are designed to support tabular data and focus on benchmarking based on plot visualization.

Hyndman and Khandakar [12] and Hyndman et al. [13] present frameworks for automatic forecasting using mainly statistical models such as `ARIMA` and exponential smoothing state space model (`ETS`). Hyndman and Khandakar [12] produced the well-known R-package named *forecast*, which can be used for automatic time series prediction. The R-package of Moreno, Rivas, and Godoy [22] also facilitates time series prediction using simple differencing (`diff`) and Box-Cox transform (`BCT`). Furthermore, we observed three works worth mentioning. Diebold and Mariano [7] propose various tests to compare the predictive accuracy of two different prediction models. Diebold and Lopez [6] propose an ensemble approach using different prediction models. Kumar et al. [17] propose a class of analytics systems to manage model selection using key ideas from data management research.

Besides, hyperparameters optimization is also a deeply studied subject [1,15, 16]. The studies may focus on exploring the hyperparameter search space using a certain heuristic. Conversely, some approaches target the right establishing of the hyperparameters to explore using either grid search or a more advanced search strategy. The Grid Search approach is commonly adopted to explore a broad range of hyperparameter settings. It consists of repeatedly training the learning algorithm with different possible hyperparameter settings combinations. At the end of the process, the hyperparameter setting that resulted in the lowest prediction errors (measured in a separate validation set) is chosen [34]. Such optimized hyperparameter settings can then be used to fit the learning model [28]. All these approaches try to lower global prediction error in machine learning but are resilient to the problem of data overfitting. Such an issue occurs when the fitted model is too dependent on the training dataset. One consequence is the fitted model's inability to generalize to unseen data observations [32].

All in all, several works present frameworks and tools for `MLM` performance assessment. Nonetheless, to our knowledge, no work proposes and implements a framework for the seamless integration of hyperparameter optimization of data preprocessing and time series prediction methods.

4 TSPredIT

The main modules of the TSPredIT framework are depicted in Fig. 2 as a UML class diagram. TSPredIT has five main functionality modules: *Preprocessing* (in purple), *Modeling* (in blue), *Sampling* (in yellow), *Evaluating* (in pink), and *Tuning* (in green). Together, they are used to support the time series prediction process. The colors of the classes are associated with their participation in the time series prediction process as depicted in Fig. 1.

All classes are inherited from TSBase. It provides a basic fit method and some attributes for introspection (to support provenance). It also includes a fit analysis of the data to adjust basic parameter values. A TSData class also provides a uniform perspective for time series data and its transformation to sliding windows. These two types are a specialization of TSData.

The first module is responsible for preprocessing (transform) and postprocessing (inverse transform) a time series. The model groups two main features. The first is related to data transformation. Especially it includes the implementation of the main nonstationary time series transformation methods [30], being either mapping-based, namely the logarithmic transform (`LT`), `BCT`, percentage change transform (PCT), moving average smoother (MAS), and diff, or splitting-based, such as empirical mode decomposition (EMD) and wavelet transform (WT). All these methods are implemented as a specialization of the Transformation class. The basic implements fit, transform, and inverse transform to provide the desired behavior.

Furthermore, it also groups a set of methods related to data augmentation. All methods include warping, flipping, and jittering [23,33]. These methods are used during tuning, which is explained later.

The second module is related to Sampling. It is responsible for converting a time series to sliding windows. It is also responsible for separating the data (TSData) into training and testing. The test size uses only recent observations to avoid introducing new data during training. Like augmentation, sampling is used during tuning.

Other relevant preprocessing methods for time series prediction are specialized from Preprocessing class, in purple. It includes support for handling missing values and data normalization such as Sliding Windows min-max normalization (swminmax), Min-max normalization (gminmax), diff, and Adaptive normalization (an) methods [24].

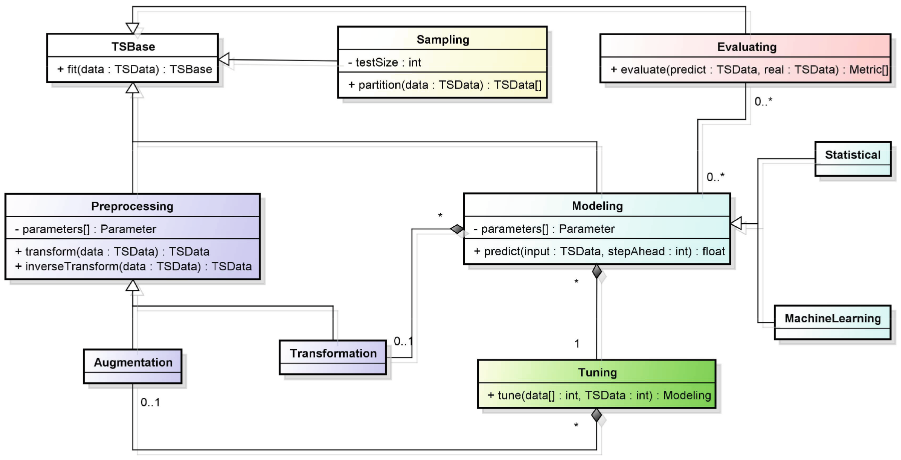
TSBase
+ fit(data : TSData) : TSBase
Sampling
- testSize : int
+ partition(data : TSData) : TSData[]
Evaluating
+ evaluate(predict : TSData, real : TSData) : Metric[]
0..*
Statistical
0..*
Preprocessing
- parameters[] : Parameter
+ transform(data : TSData) : TSData
+ inverseTransform(data : TSData) : TSData
*
Modeling
- parameters[] : Parameter
+ predict(input : TSData, stepAhead : int) : float
*
MachineLearning
1
0..1
Augmentation
Transformation
Tuning
+ tune(data[] : int, TSData : int) : Modeling
0..1
*

Fig. 2. TSPred-IT functionality modules and pre-implemented algorithms (Color figure online)

The *Modeling* module, in blue, is responsible for modeling (fit) and predicting (predict) a time series based on a particular time series prediction method. These tasks are specialized for either statistical or machine learning models. For the latter, the framework is prepared to perform any necessary machine learning life-cycle tasks during the training and prediction steps, including coercing data into sliding windows, normalizing and transforming the input data. This module includes the implementation of the statistical models: ARIMA, Holt-Winter's exponential smoothing (HW), theta forecasting (TF), and ETS. *MLM* models include multilayer perceptron network (`MLP`), random forest regression (`RFR`), support vector machine (`SVM`), `MLP`, and extreme learning machines network (`ELM`). Furthermore, the module provides deep learning models available in the *PyTorch* library, namely convolutional neural network (`Conv1D`) and long short-term memory neural network (`LSTM`). Models are associated with zero or one data preprocessing transformation and data augmentation technique. However, it might be set up with multiple candidate options chosen during tuning.

The *Tuning* module is designed to provide hyperparameter optimization. It is invoked during the fitting of a model. Hyperparameter optimization occurs whenever the modeling of time series or preprocessing transformations has a degree of freedom to adjust. The default Tuning applies time series cross-validation using the training set [14]. Data augmentation might be applied in each partition, and the model is trained after applying the data transformation. The fittest model and data preprocessing method were discovered using a grid-search, *i.e.*, the one that leads to better prediction during cross-validation, is chosen for training using the entire training set. Since this is not the only way of conducting Hyperparameter optimization, the default Tuning class can be specialized to provide other ways of enhancing this feature.

Finally, the *Evaluating* module, in pink, is responsible for assessing the model fitness and quality of predictions. These tasks are specialized for computing either prediction accuracy (error) measures or model fitting criteria. The available prediction accuracy measures include `MSE`, `sMAPE`, and maximal error. It also includes model fitness criteria such as `AIC`, `BIC`, and log-likelihood [5].

TSPredIT can integrate the described modules in a *workflow*, connecting the five modules described. The package provides the means to perform the *benchmarking* of several prediction models. It is important to remark that although providing several pre-implemented options, TSPredIT design enables the user to define and apply customized time series prediction methods.

Moreover, the package provides several automatized features for any time series prediction application. Among them, some of the main features are (i) seamless recursive combination of two or more transformation methods; (ii) seamless integration of transformation methods to the prediction process [30], which demands the combination of predictions for each component resulting from data decomposition (first package to include this approach); (iii) transformation and model parameter selection; (iv) multistep-ahead or one-step-ahead predictions; (v) rolling origin evaluation [14] for both statistical and machine learning models, and (vi) machine-learning life-cycle tasks performed during training and prediction steps. Data normalization and sliding window transformation are seamlessly conducted during machine learning model training.

The framework is implemented in R using the S3 class system [36]. TSPredIT is currently available on GitHub[1]. It is an ongoing evolution of TSPred [31], built on top of the DAL Toolbox[2].

[1] https://github.com/cefet-rj-dal/tspredit.
[2] https://cran.r-project.org/web/packages/daltoolbox/index.html.

5 Usage of TSPredIT

This section gives examples of *TSPredIT* usage. The first example corresponds to a time series prediction using a wrapper for the `MLP` model using sliding windows min-max normalization. The hyperparameter tuning applies a grid search using time series cross-validation.

The Listing 1.1 the *TSPredIT* R-package processes a time series. The lines (1–3) of code target and load the installation of *TSPredIT*. The components for the time series process can be defined separately to enable reuse. Besides, the dataset used is made available in the R-package. It is loaded using the data function (line 6).

The time series is converted into sliding windows (line 8). All sliding windows are shifted with overlap with step 1 by default. In the example, the size of the sliding windows is 8. Besides, the last 4 windows are reserved for testing (line 10), and the complement is used for testing in order to control more precisely which observations is being considered, as a fine tuning. Finally, the training data is separated into input and output (line 12).

The hyperparameter setup is established in lines 15–17. It indicates the data preprocessing option of min-max sliding windows. The input size for model building varies between 3 and 7. The base model is related to `MLP`. No data augmentation method is used in this example: $ts\_augment()$. Also, some specific parameters for `MLP` are indicated in lines 18–19. It provides ranges for the number of neurons in the hidden layer, the rate of decay during training, and the maximum number of iterations (fixed). In lines 20–21, the actual tuning is executed using the training set. Internally, it splits the data using time series cross-validation. The build model hyperparameters that work better during cross-validation are used to build the final model using the entire training set. In this example, 500 configurations were explored, each one ten times due to the default ten-fold cross-validation.

Lines 23–26 present the level of adjustment for the time series in the training set. This aspect is important since the error level in training is commonly higher during testing. It provides an expected entry error. Lines 28–31 present the prediction for testing. It applies a rolling origin with one step-ahead prediction, leading to four predictions from previously known observations. The `sMAPE` is presented in line 33.

To clarify how extensible is *TSPredIT* in providing alternative `MLM`, Listing 1.2 changes four lines of code to switch the `MLP` to `ELM`, with different ranges of hyperparameters to explore. This feature is possible due to the wrapper classes provided by TSPredIT that integrate state-of-the-art methods. Additionally, novel methods can be wrapped. Writing the fit and predict methods is needed to incorporate a novel method at *TSPredIT*.

6 Features Evaluation

TSPredIT was experimentally evaluated to expose the main features of the framework. For that, it was derived a time series dataset from public data avail-

Listing 1.1. Example of time series prediction process in TSPredIT

```
> library(daltoolbox)
#https://cefet-rj-dal.github.io/tspredit/
> library(tspredit)

# Loading fertilizers dataset
 > data(fertilizers)
# Converting to sliding windows
 > ts <- ts_data(fertilizers$brazil_n, sw = 8)
# Partitioning into training and testing
 > samp <- ts_sample(ts, test_size = 4)
# Separeting input and output for training
 > io_train <- ts_projection(samp$train)

# Hyperpararameter
 > tune <- ts_maintune(preprocess = list(
     ts_norm_swminmax()), input_size = c(3:7),
     base_model = ts_mlp(), augment = list(ts_aug_none()))
 > ranges <- list(size = 1:10, decay = seq(0, 1, 1 / 9),
         maxit = 10000)
 > model <- fit(tune, x = io_train$input,
         y = io_train$output, ranges)

# Measuring the level of adjustment
 >  adjust <- predict(model, io_train$input)
 >  ev_adjust <- evaluation.tsreg(io_train$output, adjust)
 >  print(ev_adjust$metrics$sMAPE)

# Obtaining the prediction
 > io_test <- ts_projection(samp$test)
 > prediction <- predict(model,  x = io_test$input,
         steps_ahead = 1)
 > ev_test <- evaluation.tsreg(io_test$output, prediction)
 > print(ev_test$metrics$sMAPE)
```

Listing 1.2. R example present the simplicity to explore different setups

```
# Hyperpararameter
 > tune <- ts_maintune(preprocess = list(ts_norm_swminmax()),
         input_size = c(3:7), base_model = ts_elm(),
         augment = list(ts_aug_none()))
 > ranges <- list(nhid = 1:20,
         actfun=c('sig','radbas','tribas','relu','purelin'))
 > model <- fit(tune, x = io_train$input,
         y = io_train$output, ranges)
```

able at the International Fertilizer Association (IFA)[3] as a proof of concept of the present framework. This dataset of fertilizers was explored in deep in previous work [25]. It contains data on the annual consumption of three fertilizers (K_2O, N, P_2O_5) among the top ten main consumer countries. Each time series contains 60 observations from 1961 to 2020. Observations from 1961–2016 are used for training, and observations from 2017–2020 are used for testing. For the evaluation, we selected Brazil, the third major fertilizer consumer. All coding for the experimental evaluation is available[4].

The goal of this paper is to explore multiple facets of TSPredIT. The first experiment evaluated the effect of data transformation (`swminmax`, `diff`, `an`, `gminmax`) during prediction using `MLP` as `MLM`. Figure 3 compares these methods during testing for the three fertilizers K_2O, N, and P_2O_5 in Brazil. It presents the `sMAPE` error during testing. The `swminmax` outperformed other methods for K_2O and N. However, in P_2O_5, `an` was better followed close by `swminmax`.

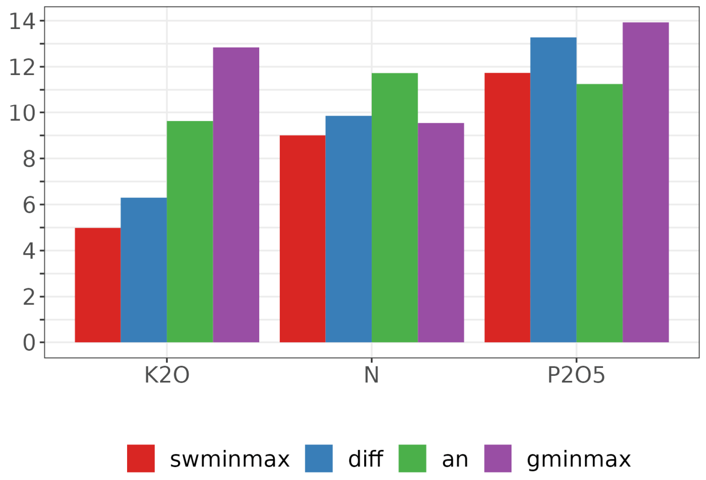

Fig. 3. Comparison of data transformations applied (`swminmax`, `diff`, `an`, `gminmax`)

A second evaluation explored the adoption of data augmentation techniques while fixing both `MLP` as `MLM` and `swminmax` as a data preprocessing technique. Figure 4 compares the performance of not applying data augmentation (none), using jittering (jitter) and warping stretching (stretch). As it can be observed, none was better both in K_2O and P_2O_5. However, for N, both jitter and stretch were worth value. The prediction performance increased by more than 1%. The data augmentation technique was seamlessly applied during time series cross-validation for training during hyperparameter optimization.

[3] http://www.fertilizer.org.
[4] https://eic.cefet-rj.br/~dal/tspredit/.

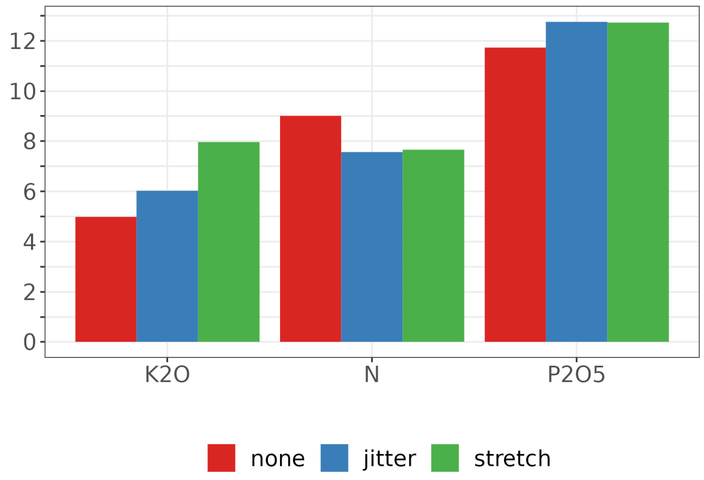

Fig. 4. Comparison of data augmentation applied (**none**, **jitter**, **stretch**)

Finally, the third evaluation studied the adoption of different `MLM` for predicting P_2O_5 using `an`. Figure 5 presents the evaluation of using `Conv1D`, `ELM`, `MLP`, `RFR`, `SVM`, `LSTM` for this scenario. Table 1 presents the hyperparameters explored.

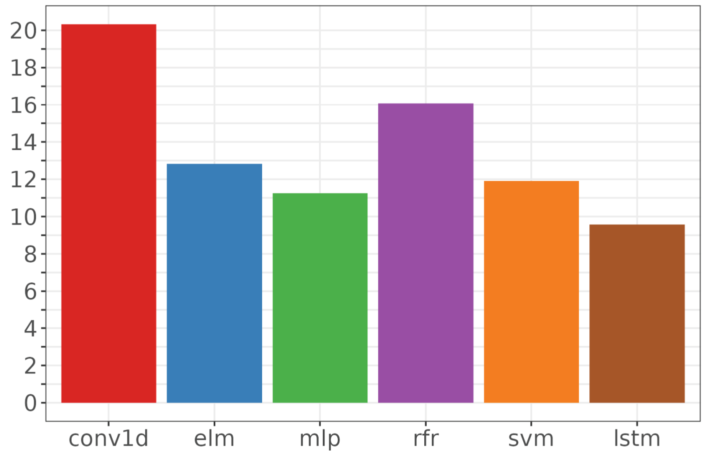

Fig. 5. Comparison of evaluation of using different `MLM` for predicting P_2O_5 using `an`

All these methods, except for `Conv1D` and `LSTM`, explored a similar amount of hyperparameter combinations (about 250 options each). The methods mostly could not improve the performance of the `MLP`. The exception was `LSTM`, which improved prediction by more than 1%.

Table 1. Used hyperparameters for each MLM

MLM	Hyperparameters
Conv1D	$input\_size \in \{3, \ldots, 7\}$
ELM	$input\_size \in \{3, \ldots, 7\}$, $nhid \in \{1, \ldots, 20\}$, $actfun \in \{$"sig", "radbas", "tribas", "relu", "purelin"$\}$
LSTM	$input\_size \in \{3, \ldots, 7\}$
MLP	$input\_size \in \{3, \ldots, 7\}$, $size \in \{1, \ldots, 10\}$, $decay \in seq(0, 1, 1/9)$
RFR	$input\_size \in \{3, \ldots, 7\}$, $nodesize \in \{5, \ldots, 10\}$, $ntree \in \{1, \ldots, 10\}$
SVM	$input\_size \in \{3, \ldots, 7\}$, $kernel =$"radial", $epsilon = seq(0, 1, 0.1)$, $cost = seq(20, 100, 20)$

As a proof of concept, these results can explore the capability of TSPredIT in combing a broad range of data preprocessing techniques and state-of-the-art MLM. Besides, the framework can provide hyperparameter optimization exploring both features, aiding the selection choice for these methods.

7 Conclusions

This paper presented TSPredIT, which extends features presented in TSPred [31]. It automates the entire time series prediction process by supporting hyperparameter optimization that combines time series data preprocessing and machine learning tuning. The architecture of TSPredIT provides five main modules. It includes data preprocessing, modeling support, prediction evaluation, and model tuning. Together, they are used to support the time series prediction process. It is made available as an extended version of DAL Toolbox Package at GitHub[5].

The combination of time series transformation methods in prediction with decomposed time series, transformation and model parameter selection, multi-step or one-step-ahead prediction, rolling origin evaluation, and the management of sliding windows is a key differentiation for TSPredIT. Several benchmark datasets from time series prediction competitions come bundled with TSPredIT. This new version enables users to practice data transformation and prediction methods, gaining confidence in the developed prediction models. Besides, the framework was designed to enable users to implement their customized methods. For example, both LSTM and Conv1D were added to TSPredIT as customized methods implemented in Python. Future updates will expand the range of implemented preprocessing methods, MLM, and evaluation metrics, especially empowering the hyperparameter selection targeting choosing combinations of data preprocessing and MLM that led to more stable models. This work leaves room for

[5] https://cefet-rj-dal.github.io/tspredit/.

future implementation including multivariate time series, cost of computation time, and short and long term prediction in classical problems.

Acknowledgements. The authors thank CNPq, CAPES (finance code 001), and FAPERJ for partially sponsoring this research.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
2. Bischl, B., et al.: mlr: machine learning in R. J. Mach. Learn. Res. **17**(170), 1–5 (2016)
3. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. Wiley, Hoboken (2015)
4. Cheng, C., et al.: Time series forecasting for nonlinear and non-stationary processes: a review and comparative study. IIE Trans. (Ins. Ind. Eng.) **47**(10), 1053–1071 (2015). https://doi.org/10.1080/0740817X.2014.999180
5. Davydenko, A., Fildes, R.: Measuring forecasting accuracy: the case of judgmental adjustments To SKU-level demand forecasts. Int. J. Forecast. **29**(3), 510–522 (2013). https://doi.org/10.1016/j.ijforecast.2012.09.002
6. Diebold, F., Lopez, J.: 8 Forecast evaluation and combination. Handb. Stat. **14**, 241–268 (1996). https://doi.org/10.1016/S0169-7161(96)14010-4
7. Diebold, F., Mariano, R.: Comparing predictive accuracy. J. Bus. Econ. Stat. **20**(1), 134–144 (2002). https://doi.org/10.1198/073500102753410444
8. Eugster, M.J.A., Leisch, F.: Bench plot and mixed effects models: first steps toward a comprehensive benchmark analysis toolbox. In: Brito, P. (ed.) Compstat 2008, pp. 299–306. Physica Verlag, Heidelberg, Germany (2008)
9. Garcia, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining. Springer (aug 2014). https://doi.org/10.1007/978-3-319-10247-4
10. Gujarati, D.N.: Essentials of Econometrics. SAGE (sep 2021)
11. Hao, J., Ho, T.: Machine learning made easy: a review of Scikit-learn package in python programming language. J. Educ. Behav. Stat. **44**(3), 348–361 (2019). https://doi.org/10.3102/1076998619832248
12. Hyndman, R., Khandakar, Y.: Automatic time series forecasting: The forecast package for R. J. Stat. Softw. **27**(3), 1–22 (2008). https://doi.org/10.18637/jss.v027.i03
13. Hyndman, R., Koehler, A., Snyder, R., Grose, S.: A state space framework for automatic forecasting using exponential smoothing methods. Int. J. Forecast. **18**(3), 439–454 (2002). https://doi.org/10.1016/S0169-2070(01)00110-8
14. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (may 2018)
15. Izaú, L., et al.: Towards robust cluster-based hyperparameter optimization. In: Anais do Simpósio Brasileiro de Banco de Dados (SBBD), pp. 439–444. SBC (sep 2022). https://doi.org/10.5753/sbbd.2022.224330
16. Khalid, R., Javaid, N.: A survey on hyperparameters optimization algorithms of forecasting models in smart grid. Sustain. Cities Soc. **61**, 102275 (2020). https://doi.org/10.1016/j.scs.2020.102275
17. Kumar, A., McCann, R., Naughton, J., Patel, J.M.: Model selection management systems: the next frontier of advanced analytics. ACM SIGMOD Rec. **44**(4), 17–22 (2016). https://doi.org/10.1145/2935694.2935698

18. Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans. Softw. Eng. **34**(4), 485–496 (2008). https://doi.org/10.1109/TSE.2008.35
19. Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **379**(2194), 20200209 (2021). https://doi.org/10.1098/rsta.2020.0209
20. Lindemann, B., Müller, T., Vietz, H., Jazdi, N., Weyrich, M.: A survey on long short-term memory networks for time series prediction. In: Procedia CIRP. vol. 99, pp. 650–655 (2021). https://doi.org/10.1016/j.procir.2021.03.088
21. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: a review. IEEE Trans. Knowl. Data Eng. **31**(12), 2346–2363 (2019). https://doi.org/10.1109/TKDE.2018.2876857
22. Moreno, A.V., Rivas, A.J.R., Godoy, M.D.P.: predtoolsTS: Time Series Prediction Tools. Tech. rep.,https://cran.r-project.org/package=predtoolsTS (2018)
23. Mumuni, A., Mumuni, F.: Data augmentation: a comprehensive survey of modern approaches. Array **16**, 100258 (2022). https://doi.org/10.1016/j.array.2022.100258
24. Ogasawara, E., Martinez, L., De Oliveira, D., Zimbrão, G., Pappa, G., Mattoso, M.: Adaptive normalization: a novel data normalization approach for non-stationary time series. In: Proceedings of the International Joint Conference on Neural Networks (2010). https://doi.org/10.1109/IJCNN.2010.5596746
25. Pacheco, C., et al.: Exploring data preprocessing and machine learning methods for forecasting worldwide fertilizers consumption. In: Proceedings of the International Joint Conference on Neural Networks. vol. 2022-July (2022). https://doi.org/10.1109/IJCNN55064.2022.9892325
26. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
27. Ramey, J.A.: sorting hat: sorting hat. Tech. rep., https://cran.r-project.org/web/packages/sortinghat/index.html (2013)
28. Ran, Z.Y., Hu, B.G.: Parameter identifiability in statistical machine learning: a review. Neural Comput. **29**(5), 1151–1203 (2017). https://doi.org/10.1162/NECOa00947
29. Salles, R., Assis, L., Guedes, G., Bezerra, E., Porto, F., Ogasawara, E.: A framework for benchmarking machine learning methods using linear models for univariate time series prediction. In: Proceedings of the International Joint Conference on Neural Networks. vol. 2017-May, pp. 2338–2345 (2017). https://doi.org/10.1109/IJCNN.2017.7966139
30. Salles, R., Belloze, K., Porto, F., Gonzalez, P., Ogasawara, E.: Nonstationary time series transformation methods: an experimental review. Knowl.-Based Syst. **164**, 274–291 (2019). https://doi.org/10.1016/j.knosys.2018.10.041
31. Salles, R., Pacitti, E., Bezerra, E., Porto, F., Ogasawara, E.: TSPred: a framework for nonstationary time series prediction. Neurocomputing **467**, 197–202 (2022). https://doi.org/10.1016/j.neucom.2021.09.067
32. Sarwar Murshed, M., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., Hussain, F.: Machine learning at the network edge: a survey. ACM Comput. Surv. **54**(8), 1–37 (2022). https://doi.org/10.1145/3469029
33. Talavera, E., Iglesias, G., González-Prieto, A., Mozo, A., Gómez-Canaval, S.: Data Augmentation techniques in time series domain: A survey and taxonomy (jun 2022). https://doi.org/10.48550/arXiv.2206.13508,http://arxiv.org/abs/2206.13508
34. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007). https://doi.org/10.1007/s11222-007-9033-z

35. Wen, Q., et al.: Time series data augmentation for deep learning: a survey. In: IJCAI International Joint Conference on Artificial Intelligence, pp. 4653–4660 (2021)
36. Wickham, H.: Advanced R. CRC Press, second edn. (may 2019)