Electrical Power and Energy Systems 155 (2024) 109579

Contents lists available at ScienceDirect

International Journal of Electrical Power and Energy Systems

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

# Optimized hybrid ensemble learning approaches applied to very short-term load forecasting

Marcos Yamasaki Junior [a], Roberto Zanetti Freire [b], Laio Oriel Seman [c], Stefano Frizzo Stefenon [d,e], Viviana Cocco Mariani [f,g,*], Leandro dos Santos Coelho [a,g]

[a] Industrial and Systems Engineering Graduate Program, Pontifical Catholic University of Parana (PUCPR), Curitiba 80215-901, Brazil
[b] Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba 80230-901, Brazil
[c] Department of Automation and Systems Engineering, Federal University of Santa Catarina (UFSC), Florianopolis 88040-900, Brazil
[d] Digital Industry Center, Fondazione Bruno Kessler, Trento 38123, Italy
[e] Department of Mathematics, Computer Science and Physics, University of Udine, Udine 33100, Italy
[f] Mechanical Engineering Graduate Program, Pontifical Catholic University of Parana (PUCPR), Curitiba 80215-901, Brazil
[g] Department of Electrical Engineering, Federal University of Parana (UFPR), Curitiba 81530-000, Brazil

ARTICLE INFO

*Keywords:*
Electrical power systems
Ensemble learning
Machine Learning
Short-term load forecasting
Signals decomposition methods
Time series forecasting

ABSTRACT

The significance of accurate short-term load forecasting (STLF) for modern power systems' efficient and secure operation is paramount. This task is intricate due to cyclicity, non-stationarity, seasonality, and nonlinear power consumption time series data characteristics. The rise of data accessibility in the power industry has paved the way for machine learning (ML) models, which show the potential to enhance STLF accuracy. This paper presents a novel hybrid ML model combining Gradient Boosting Regressor (GBR), Extreme Gradient Boosting (XGBoost), k-Nearest Neighbors (kNN), and Support Vector Regression (SVR), examining both standalone and integrated, coupled with signal decomposition techniques like STL, EMD, EEMD, CEEMDAN, and EWT. Through Automated Machine Learning (AutoML), these models are integrated and their hyperparameters optimized, predicting each load signal component using data from two sources: The National Operator of Electric System (ONS) and the Independent System Operators New England (ISO-NE), boosting prediction capacity. For the 2019 ONS dataset, combining EWT and XGBoost yielded the best results for very short-term load forecasting (VSTLF) with an RMSE of 1,931.8 MW, MAE of 1,564.9 MW, and MAPE of 2.54%. These findings highlight the necessity for diverse approaches to each VSTLF problem, emphasizing the adaptability and strength of ML models combined with signal decomposition techniques.

## 1. Introduction

Electricity is essential for the development of humanity, and its use is growing in different areas such as residential and commercial buildings, industry, medicine, transportation, public lighting, robotics and machinery, electro-valves, refrigeration, and air conditioning equipment, among others [1]. The energy resulting from electricity is present in different sectors, and often machines can work 24 h a day, non-stop, due to electricity. Electricity is non-storable and requires a stable electrical system with a constant balance between production, transmission, and demand [2]. There are many types of electricity generation units, such as thermal energy, which is converted into electricity, hydroelectric energy, which is converted by gravitational potential or kinetic energy from a hydro-power source, photovoltaic energy, which is converted from solar energy using solar cells, wind energy which is converted from mechanical energy using wind turbines [3].

Energy use increases with per capita Gross Domestic Product (GDP) analysis; richer countries usually consume more energy per person than developing countries. Therefore, this economic relationship is also based on how much energy the country can produce and how much energy it will need to produce in the coming years, so countries must do planning and invest in the infrastructure needed for energy generation. The power utilities or power plant companies depend on the estimated electricity demand to meet the load connected to the grid [4].

Regarding the forecast horizon, Short Term Load Forecasting (STLF) can be useful for real-time applications such as controlling electric power generation units [5]. The medium-term forecast can be used for resource budgeting, and the long-term forecast can be used to plan an

* Corresponding author.
*E-mail addresses:* marcos.yamasaki@siemens.com (M. Yamasaki Junior), robertofreire@utfpr.edu.br (R.Z. Freire), laio@ieee.org (L.O. Seman), sfrizzostefenon@fbk.eu (S.F. Stefenon), viviana.mariani@pucpr.br (V.C. Mariani), leandro.coelho@pucpr.br (L. dos Santos Coelho).

https://doi.org/10.1016/j.ijepes.2023.109579
Received 31 March 2023; Received in revised form 20 July 2023; Accepted 13 October 2023
Available online 26 October 2023
0142-0615/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

expansion of the power system network. However, the electricity demand depends on weather conditions such as temperature, wind speed, precipitation, and other factors, and the demand changes depending on the day, season, trade, and industry activities such as peak hours, normal hours, weekdays, weekend days, holidays, and others. Thus, seasonality, trend, noise, outliers, and other aspects are present when power demand is studied, demanding from power supply companies accurate load forecasting techniques to decrease electricity production losses and costs [6].

Electricity time series data generally has nonlinear behavior, which means random and periodic components embedded in the series, such as time, seasonal events, economic activity behavior, measurement errors (outliers), missing values, and noise [7]. These components are intrinsic to the time series data, and each data set will contain different features. Then, pre-processing and data transformation steps are required to provide appropriate inputs for the forecasting models. With proper methodology, Machine Learning (ML) models can handle such adversities, accurately predicting the next steps to be taken in forecasting tasks [8].

ML is a sub-area of Artificial Intelligence (AI) that provides applied models in classification, regression, and prediction that learn from experience. Experience comes from experimentally measured historical data; such data can be text, audio, images, videos, examinations in the field of medicine, sensor measurements, and electricity consumption, among other available data sources [9]. ML models can learn from experience and use this knowledge to reproduce some tasks, such as predicting energy load consumption a week in advance, predicting weather conditions, preventing failures in machines that are in operation, etc. Over the load forecasting horizon, the number of steps ahead to be predicted can be determined in four different categories, Very Short Term Load Forecasting (VSTLF), STLF, Medium Term Load Forecasting (MTLF), and Long Term Load Forecasting (LTLF) [10]. The cut-off horizons for these four categories are one day, two weeks, and three years, respectively.

Besides the advances in AI and ML applications, many challenges still exist to overcome. The training for some ML models is sometimes ineffective and superficial, requiring more computational effort and generalization capacity [11]. The ML models do not have their parameters automatically adjusted, called hyperparameters, which directly affect their performance. Some recent studies, such as the one presented in [12], indicated that combining various individual ML models can produce a better result when compared with individual ones when built properly. Additional works addressing the previously mentioned aspects are presented in the sequence.

The ensemble models proved to be better than its worst individual predictor and, in some cases, better than its best individual predictor. Wang et al. [13] applied an adaptive decomposition method based on Variational Mode Decomposition (VMD) and SampEn to decompose the data and the eXtreme Gradient Boosting (XGBoost) for short-term load forecasting, which performed better than any individual learning algorithm. This work uses two different datasets to investigate ensemble model and decomposition techniques for VSTLF of one day ahead. The main contributions of this research are summarized as follows:

(i) ML models, including Gradient Boosting Regressor (GBR), XGBoost, k-Nearest Neighbors (kNN), and Support Vector Regression (SVR) models are designed and evaluated in both individual and combined models for VSTLF.
(ii) A new hybrid framework based on GBR, XGBoost, kNN, and SVR integrated with signal decomposition methods like Seasonal-Trend decomposition using locally estimated scatterplot smoothing, LOESS (STL), Empirical Mode Decomposition (EMD), Ensemble Empirical Mode Decomposition (EEMD), Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN), and EWT is developed to enhance the accuracy of classical ML structures in VSTLF.
(iii) The hyperparameters tuning of the ML models integrated with decomposition methods was based on an automated machine learning method.
(iv) An extensive evaluation based on two datasets, one from the ONS and the other from the Independent System Operators New England (ISO-NE), is performed to demonstrate the potential of the proposed ML integrated with decomposition methods in VSTLF. Pre-processing steps of data cleaning, feature engineering, and outlier processing were considered.
(v) Well-established metrics for VSTLF as RMSE, Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) with 10-fold cross-validation were adopted to evaluate the models. The statistical tests demonstrated that the proposed models provided superior performance using signals decomposition compared to ML models without the time series signals decomposition process in VSTLF.

The experiments of VSTLF were split into two datasets, ONS and ISO-NE. The objective was to evaluate how well the regression models may predict the electricity demand between different ML algorithms with and without decomposition techniques. To achieve the lowest values for RMSE and MAPE performance metrics for all evaluated ML models, an AutoML toolkit has been used to select the best hyperparameters.

The remainder of this study is organized as follows. Related works focusing on forecasting models using ML were described in Section 2. In Section 3, the forecasting models are presented. The framework adopted in this study is shown in Section 4. The two datasets, the main results obtained together with the choice of hyper-parameters and their justifications, are presented in Section 5. Finally, the last section presents the conclusion and indicates steps for future work.

## 2. Related works

This section briefly presents short-term power system load forecasting models in the specific literature field. Both EWT and improved density-based spatial clustering (IDBSCAN) models were used to forecast applications with noise in Zhang and Zhang [14]. The EWT decomposes the load into intrinsic mode functions (IMFs), which are predicted using rational models and long-term memory (LSTM). The IDBSCAN and meteorological factors group the high-frequency components, which are overlaid to obtain the complete time series forecasting results.

Sankalpa et al. [15] considered voting regression (VR) for short-term load forecasting assuming five distinct models, i.e., three parametric multiple linear regressors and two non-parametric machine-learning models. The cross-validation (CV) procedure selects models, and the Blocked-CV technique yields the closest validation error to the test error. Assuming the VR method, the strategy outperformed the individual predictions of the models. The LSTM model and penalized quantile regression were used to forecast daily and weekly indoor loads in Duan [16]. The suggested technique outperforms some classical models in coverage probability by 6.4% to 20.9%.

Self-attention-based short-term load forecasting (STLF) employs non-parametric kernel density estimation to create customer electricity consumption feature curves, variational modal decomposition, and a maximum information coefficient for feature selection in Yu et al. [17]. The AdaBlief optimizer was applied to obtain the model parameters, and an Informer based on increased self-attention predicts intrinsic mode function components. The proposed STLF outperforms other models. Baliyan et al. [18] examined short-term load forecasting using hybrid neural networks, which combine neural networks and stochastic learning methods, including genetic algorithms and particle swarm optimization. Short-term load forecasting is crucial to power system efficiency and reliability.

In Groß et al. [19], it is noticed that decentralized flexibilities must be managed more efficiently as renewable production becomes increasingly volatile. The research eight-day-ahead electricity load forecasting approaches for supermarkets, schools, and houses. Machine learning, statistics, and a median ensemble of forecasts were investigated. Compared to a naïve seasonal benchmark approach, nearly all strategies improved predicting accuracy by up to 35%. It concludes that finding the appropriate load forecasting strategy is a task-specific challenge.

The research from Yu et al. [20] presented a graph representation learning-based short-term load forecasting approach. The load graph encodes electricity use and the coronavirus disease of 2019 effects to anticipate future loads. A residual graph convolutional network models the non-linear correlations between the graph and future loads, and a graph concatenation model improves learning efficiency. The epidemic strongly correlates with regional electricity use, and the strategy outperforms other representative methods.

Sun et al. [21] proposed a study from Tai'an, Shandong Province, China's hourly historical loads and show significant daily and weekly fluctuations. According to the authors, the Chinese Lunar Spring Festival and other factors affect load. A six-periodic, three-nonperiodic artificial neural network model is created to address this. The model was trained on data from January 2016 to August 2018 and showed that the daily prediction model with specified parameters can improve predicting accuracy.

Genov et al. [22] proposed one and three-day-ahead load forecasting in smart grids using feed-forward artificial neural networks, recurrent neural networks, and cross-learning algorithms. To test those strategies, they used high-resolution multi-seasonal electricity demand data from buildings in Belgium, Canada, and UK. The optimal model depends on the accuracy metric, but both feedforward and recurrent neural network models perform well.

A clustering-based filter feature selection strategy to improve short-term load forecasting models is proposed in Subbiah and Chinnappan [23]. A recurrent neural network-based LSTM model for short-term load forecasting was compared to Multilayer Perceptron, Radial Basis Function, Support Vector Regression, and Random Forest. Fast Correlation Based Filters (FCBF), Mutual Information, and RReliefF (Relief for Regression) were evaluated to decrease the curse of dimensionality and increase performance — clustering groups' load patterns, and eliminate outliers. LSTM with RReliefF outperforms other models on two European datasets.

Candela Esclapez et al. [24] suggested automating variable processing and selection to increase forecasting accuracy and interpretability. The dataset for peninsular demand from Spanish energy company (*Red Eléctrica de España*) was tested, finding that it reduces forecasting error by 0.16% in MAPE and 59.71 MWh in RMSE. The authors observed that heat affects consumption more than cold, and on hot days the temperature of the second prior day affects consumption more than the preceding one. The LSTM is currently widely used for time series forecasting due to its ability to deal with nonlinearities [25], which can be tail oscillations caused by abrupt variations in time series, resulting in higher frequencies in the signal and requiring more specialized models.

The reviewed works show that many models have been employed to address the power system load forecasting; a summary of the applied methods covered in this section, considered evaluation metrics, case of study, and reference are presented in Table 1.

## 3. Employed methods

This study uses the AutoML framework to analyze various decomposition and regressor models. The best models are chosen based on their performance on the validation dataset. This choice is made even if the best model is a single regressor or a combination.

This section outlines the main decomposition techniques and regression models used in this research. The final model selection was performed using AutoML, which evaluated the performance of various combinations of models and picked the best-performing one. This approach reduces the need for manual intervention and enhances performance by combining the strengths of different models.

### 3.1. Decomposition techniques

This subsection delves into the analysis of decomposition techniques applicable to time series analysis evaluated in this paper. These techniques include Decomposition using Locally Estimated Scatterplot Smoothing, Empirical Mode Decomposition, Ensemble Empirical Mode Decomposition, Complete Ensemble Empirical Mode Decomposition with Adaptive Noise, and Empirical Wavelet Transform.

#### 3.1.1. Decomposition using Locally Estimated Scatterplot Smoothing (STL)

STL is a nonparametric and flexible method of decomposing a time series [26]. The decomposition model can be written as:

$$Y(t) = T(t) + S(t) + R(t) \tag{1}$$

where $Y(t)$ is the observed series, $T(t)$ is the trend component, $S(t)$ is the seasonal component, and $R(t)$ is the remainder.

#### 3.1.2. Empirical Mode Decomposition (EMD)

EMD is a method that identifies and separates the IMFs in a time series [27]. The decomposed time series can be represented as:

$$Y(t) = \sum_{i=1}^{n} IMF_i(t) + R_n(t) \tag{2}$$

where $Y(t)$ is the observed series, $IMF_i(t)$ is the $i$th intrinsic mode function, and $R_n(t)$ is the residual.

#### 3.1.3. Ensemble EMD (EEMD)

EEMD is an enhancement of the EMD approach, which includes a noise-assisted data analysis method [28]. This allows extracting more accurate and true intrinsic mode functions (IMFs). The EEMD decomposition model is similar to the EMD model, and can be written as:

$$Y(t) = \sum_{i=1}^{N} \frac{1}{M} \sum_{j=1}^{M} IMF_{ij}(t) + R_N(t) \tag{3}$$

in which $Y(t)$ is the original series, $N$ is the total number of IMFs, $M$ is the total number of trials with added white noise, $IMF_{ij}(t)$ represents the $j$th trial IMF for the $i$th mode, and $R_N(t)$ is the final residual.

### 3.2. Complete ensemble EMD with Adaptive Noise (CEEMDAN)

CEEMDAN further improves upon EEMD by adapting the noise level to the signal characteristics [29]. It reduces mode mixing and enhances the extraction of true and more physically meaningful IMFs. The decomposition model remains similar to the EMD and EEMD models.

### 3.3. Empirical Wavelet Transform (EWT)

EWT is an alternative to the standard wavelet transform, which better adapts to the signal characteristics [30]. The decomposition model can be represented as:

$$Y(t) = \sum_{i=1}^{n} W_i(t) + R_n(t) \tag{4}$$

where $Y(t)$ is the observed series, $W_i(t)$ is the $i$th wavelet function, and $R_n(t)$ is the residual.

These techniques are instrumental in performing comprehensive time series analysis, extracting underlying patterns, and improving predictive modeling. Each technique has its own set of advantages and

**Table 1**
Literature models to deal with the short-term load forecasting problem.

| Technique | Error metric | Case/Dataset | Reference |
|---|---|---|---|
| EWT and IDBSCAN | MAPE, RMSE | Chinese city | [14] |
| VR-based Ensemble Learning | MAPE | Electricity Generating Authority of Thailand | [15] |
| Long Short-Term Memory and Penalized Quantile Regression | RMSE, MAE, CVRMSE[a] | Office building Shanghai, China | [16] |
| STLF with feature selection based considering VMD | MAPE, RMSE | Spanish regional level | [17] |
| Hybrid Neural Networks with stochastic learning techniques | MAPE | Review based on distinct datasets | [18] |
| Ensemble of Machine Learning Models and statistics | NRMSE[b] | Residential buildings, schools, and supermarkets in Germany | [19] |
| Residual Graph Convolutional Network | MAPE, RMSE | Historical load information from major USA power markets | [20] |
| Artificial Neural Networks and features selection | MAE, MPE,[c] MAPE | Tai'an, Shandong Province, China | [21] |
| Recurrent Neural Networks and Cross-learning algorithms | RMSE, MAPE | Buildings in England, Canada, Belgium, and Green Energy Park | [22] |
| LSTM and feature selection FCBF, Mutual Information, and RReliefF | MSE,[c] RMSE | Historical load and weather data of Switzerland and France | [23] |
| Exogenous AutoRegressive (EAR) model and group of EAR networks | MAPE, RMSE | Spanish electricity operator | [24] |
| Wavelet transform and LSTM with attention mechanism | MSE, MAE, MAPE | Hydroelectric power plant in Brazil | [25] |

[a] CVRMSE — Coefficient of Variance of the Root Mean Squared Error.
[b] NRMSE — Normalized Root Mean Squared Error.
[c] MPE — Mean Percentage Error.
[d] MSE — Mean Squared Error.

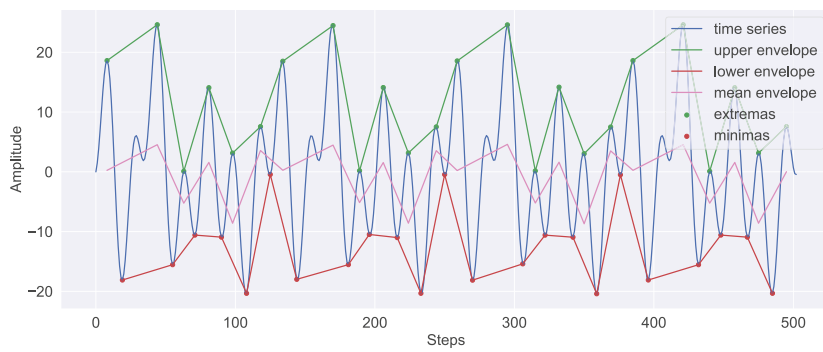



**Fig. 1.** Trend signal decomposition example.

constraints, and their application depends on the nature of the time series. Fig. 1, using the EMD method, presents an example of how the decomposition is performed. Since all decomposition used here is focused on the trend, similar results would be found using different techniques. In this paper, the mean envelope is considered.

### 3.4. Regression models

This subsection presents the regression models applied to perform time series forecasting.

#### 3.4.1. Autoregressive Moving Average (ARMA)

Autoregressive Moving Average (ARMA) models provide a weakly stationary stochastic time series in terms of two polynomials, the first is the AutoRegressive (AR) model, and the second is the Moving Average (MA) model [31]. $AR(p)$ is written in Eq. (5) and $MA(q)$ is written in Eq. (6):

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \epsilon_t, \tag{5}$$

where $p$ is the order of AR model, $\varphi_1, \ldots, \varphi_p$ are parameters, $c$ is a constant, and the random variable $\epsilon_t$ is the white noise. The $MA(q)$ is given by:

$$X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}, \tag{6}$$

where $q$ is the order of MA model, the $\theta_1, \ldots, \theta_q$ are the parameters of the model, $\mu$ is the expectation of $X_t$ (usually considered equal to zero), and $\epsilon_t, \epsilon_{t-i}$ are white noises.

Thus, $ARMA(p,q)$ model is written with $AR(p)$ and $MA(q)$ as follow:

$$X_t = c + \epsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}. \tag{7}$$

The AR, MA, ARMA, and similar models are used to forecast the observation at one step-ahead $(t+1)$, based on historical data for the same time series. Some requirements must be fulfilled, such as identifying stationarity, seasonality, and outliers. The autoregressive integrated moving average model is the variation of ARMA to support differencing the time series to make it stationary, where a combination of several differences was applied to the model [32].

#### 3.4.2. Random forests

Random Forests (RF), or random decision forests, are ensemble learning models for classification, regression, and other tasks. This method operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) [33], or prediction (regression) [34], of the individual trees. The RF builds multiple decision trees and merges them for a more accurate and stable prediction. The model typically uses a subset of the input features to split each node in the decision tree, which leads to better performance and helps to avoid overfitting [35].

#### 3.4.3. Gradient Boosting regressor

Gradient Boosting is an ML model for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model stage-wise like other boosting models do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function [36].

The idea of the GBR approach is that boosting can be interpreted as an optimization algorithm on a suitable cost function. The algorithm optimizes a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in machine learning and statistics fields beyond regression and classification [37].

#### 3.4.4. Extreme Gradient Boosting (XGBoost)

XGBoost is an open-source framework, available in several programming languages, often used in ML global competitions. It is one of the best GBoost tree implementations currently available. Its parallel tree-boosting capabilities make it way faster than other tree-based ensemble methods [38]. It was designed utilizing the principles of GBoost, joining weak learners into strong learners. However, compared with gradient boosting built sequentially, slowly learning from data to improve its prediction in the next iteration, XGBoost builds trees in parallel processing [39].

#### 3.4.5. k-Nearest Neighbor regression (kNN)

The kNN works by predicting the target by local interpolation of the targets associated with the nearest neighbors in the training set, which means the nearest values of training set data may contribute more or less depending on the euclidean distance of the sample data associated with the target, which is commonly called as a query. The kNN is being used for recommendation systems, financial market prediction, and text categorization, among other fields [40].

#### 3.4.6. Support Vector Regression (SVR) and Support Vector Machine (SVM)

The SVR is a supervised learning model that can be used to solve classification and regression problems [41]. An Support Vector Machine (SVM) training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. SVM maps training examples to points in space to maximize the gap width between the two categories. New examples are then mapped into that space and predicted to belong to a category based on which side of the gap they fall. In addition to performing linear classification, SVM can efficiently perform a non-linear classification using the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces [42].

### 3.5. Stacking

The Automatic Relevance Determination (ARD) model [43], also known as Sparse Bayesian Learning or Relevance Vector Machine, is a type of Bayesian linear regression. It imposes a prior distribution on the weights, leading to automatic sparsity and relevance determination of features.

Let us denote the design matrix (with $n$ samples and $m$ features) as $X \in \mathbb{R}^{n\times m}$ and the corresponding target values as $\mathbf{y} \in \mathbb{R}^n$. We also denote the weight vector as $\mathbf{w} \in \mathbb{R}^m$. The linear model can be represented as:

$$\mathbf{y} = X\mathbf{w} + \epsilon \tag{8}$$

where $\epsilon$ is the noise term, assumed to follow a Gaussian distribution with zero mean and variance $\sigma^2$, i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2)$. In ARD, each weight $w_i$ in $\mathbf{w}$ is assumed to follow a Gaussian distribution with zero mean and its own variance $\alpha_i^{-1}$:

$$w_i \sim \mathcal{N}(0, \alpha_i^{-1}) \tag{9}$$

The ARD model aims to find the Maximum a posteriori (MAP) estimate of the weights, $\mathbf{w}^*$, and the hyperparameters $\boldsymbol{\alpha}^*$ and $\sigma^{*2}$:

$$\mathbf{w}^*, \boldsymbol{\alpha}^*, \sigma^{*2} = \arg\max_{\mathbf{w},\boldsymbol{\alpha},\sigma^2} p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{y}) \tag{10}$$

This maximization problem can be solved iteratively using Expectation–Maximization or similar optimization algorithms. When $\alpha_i$ becomes very large, the corresponding weight $w_i$ is pushed towards zero, leading to automatic sparsity. Hence, the ARD model can automatically determine the relevance of each feature and achieve sparsity in the weight vector, making it a useful meta-learner in stacking regression.

In stacking, multiple base regressor models are trained to predict the same output, and then a meta-learner model, such as the ARD model, is used to find the optimal way to combine these predictions into a final predicted output. Each base regressor, such as GBR, XGB, kNN, or SVR in the discussed framework, produces its prediction for the output. These individual predictions are then used as input features for the ARD model. In turn, The ARD model learns how to combine these individual predictions best to produce a final output that minimally deviates from the target.

One key advantage of the ARD model is its ability to perform automatic relevance determination, meaning it can evaluate the importance of each base regressor's prediction. The ARD model does this by assigning a weight to each base regressor's prediction in the final combined output, determining how much each base regressor's prediction should contribute to the final output. If a particular regressor's prediction is not contributing significantly to the accuracy of the final output, the ARD model assigns it a weight close to zero, effectively ignoring it.

This feature of the ARD model can be handy when one has many base regressors and wants to avoid overfitting by not relying too heavily on any one regressor. It also allows for more interpretable results, as it is possible to observe which base models the ARD model deemed most useful in making its predictions.

## 4. Model framework

The proposed framework presented in this study, as depicted in Fig. 2, is a six-stage process, wherein the stages are in different colors. These stages are:

- The first stage is regarding data selection and preprocessing. This paper considers two datasets: ISO-NE and ONS, from 2015 to 2019. For the preprocessing, the data is cleaned (removing unused and unnecessary values from the dataset), the date and time are decoupled into more features or entries for the model, and outliers are evaluated. The transformations (Box–Cox and StandardScaler) are applied to avoid non-linear terms and bias in the forecasting results.
- The second stage involves selecting a decomposition method from a pool of methods, including STL, EMD, EEMD, CEEMDAN, EWT, or none. The decomposition is applied to denoise the signal. This stage employs a hybrid approach, combining trend decomposition with the forecasting model.
- The third stage involves splitting the decomposed data into IMFs and residues. Organize the considering k-fold cross-validation and the setup of the hyperparameters for tuning.
- In the fourth stage, one or more regressor algorithms are selected for prediction for each IMF and residue. The options include GBR, XGB, kNN, and SVR. If more than one regressor is selected, the Stacking ensemble regressor combines the different models, with the ARD model serving as the meta-learner.
- In the fifth stage, the predictions from the various intrinsic mode functions are summed, resulting in decomposed data for the predicted data only. The transformations that were applied in the first stage are reverted.
- In the sixth stage, the models are tested on a separate test set to evaluate their performance. Each IMF has one model predictor. Finally, the performance of the models is assessed using error metrics, including RMSE, MAE, and MAPE.

During the preprocessing, the removed values include price, the energy component of real-time locational marginal price (LMP), and day-ahead LMP, among others. Additionally, null or invalid values are replaced by linearly interpolating the nearest values. Additionally, the year, month, day, weekday, holiday, and hour are separated into different features, and outliers are identified by the LOF algorithm and replaced by linear interpolation, similar to the method used in the first sub-stage.

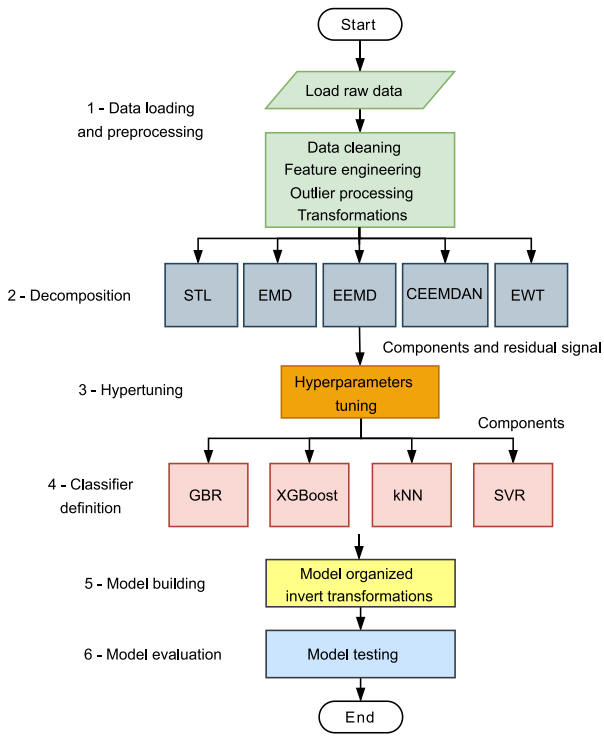



**Fig. 2.** Model framework overview.

The proposed framework provides a comprehensive, systematic approach to processing and evaluating time series data, ensuring optimal feature engineering, outlier treatment, and model selection for accurate forecasting.

### 4.1. Generalization and algorithm evaluation

Intending to provide a robust forecasting model in machine learning, it is necessary to train the model and evaluate its performance using different data and models to obtain a better response for various inputs that the model may receive to predict the output data. With a wide range of data possible, splitting train and test data and cross-validating over different data portions results in a robust forecasting model that may respond reliably for most input data, which generalizes the regression model.

#### 4.1.1. Error metrics

As it can be verified in Table 1, some o the most relevant error metrics are RMSE, MAE, and MAPE, these considered in this study, and presented in Eqs. (11) to (13). In the previously mentioned equations, $x$ is the sample, $x_i$ is the $i$th sample, $\hat{x}_i$ is the estimated sample, and $n$ is the total number of samples for training/test.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}, \tag{11}$$

$$MAE = \frac{\sum_{i=1}^{N}\left|(x_i - \hat{x}_i)\right|}{N}, \tag{12}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{(x_i - \hat{x}_i)}{x_i}\right|. \tag{13}$$

#### 4.1.2. Cross-validation

Cross-validation is one of the model validation techniques for evaluating how the results of a statistical analysis will generalize to an independent dataset. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. It is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model, and the other used to validate it. In typical cross-validation, the training and validation sets must cross over in successive rounds such that each data point has a chance of being validated. The basic form of cross-validation is $k$-fold cross-validation. There are other special cases of $k$-fold cross-validation, which may involve repeated rounds of $k$-fold cross-validation [44].

Cross-validation is often used for the evaluation or comparison of learning algorithms. In each iteration, one or more learning algorithms use $k$-folds of data to learn and adjust the weights for one or more models, and subsequently, the trained models make predictions based on the input data for the validation fold. The performance of each learning algorithm on each fold can be tracked using a set of predetermined performance metrics like accuracy or MSE. Upon completion, $k$ values of the performance metric will be available for each algorithm. Different methodologies, such as averaging, are usually used to get an aggregated result from these values. These values can be used in a statistical hypothesis test to show that one algorithm is superior to another [45].

Since the order of the data is essential, cross-validation might be problematic for time series models. A more suitable approach might be to use rolling cross-validation. However, if a single summary statistic describes the performance a stationary bootstrap may work. The bootstrap statistic needs to accept an interval of the time series and return the summary statistic [46]. The call to the stationary bootstrap must specify an appropriate mean interval length. The expanding window and sliding window methods are primarily used in this case. The training set and validation set respect the data order, increasing or sliding the training set over the dataset as shown in Figs. 3 and 4. For this research, the sliding method has been used since the results responded better when compared with other methods.

Fig. 5 corresponds to the data split of ONS dataset for model training and validation. It has used a 10-fold cross-validation split. The gray curve is the original time series, and each color represents each $k$-fold of cross-validation for the validation set. The training set comprises 130 days (3120 steps ahead), and the validation set contains 15 days (360 steps ahead) using the sliding window method.

### 4.2. Pre-processing

Before starting any regression task, one of the most important procedures is pre-processing the dataset to verify possible outliers and inconsistent data [48]. Pre-processing data might be a time-consuming task since the researcher must know every detail of the analyzed dataset and then must study how to identify outliers, what will be the policy after identifying the outlier, it should be removed or replaced, and if it is decided to replace, what will be the value replaced and so on.

#### 4.2.1. Outliers processing

Thereby, among outliers, there are many kinds of them. The usual types are point outliers, contextual outliers, or collective outliers. A point outlier is the simplest type of outlier; the single data point is far different from the rest of the distribution when compared between them. Contextual or conditional outliers can be noise in data, such as background noise signal when doing image recognition or RF noise when receiving a radio signal. It depends on the context where the dataset is induced and how it is specified as a part of the problem formulation. For instance, a power utility company usually provides between 5 GW and 10 GW during the year. For a couple of hours, it is possible to see into the dataset power measurements below 1 GW or above 50 GW, due to some sensor error or failures in the power station, which is typically not seen as possible valleys or peaks in the measurement graphics [49].

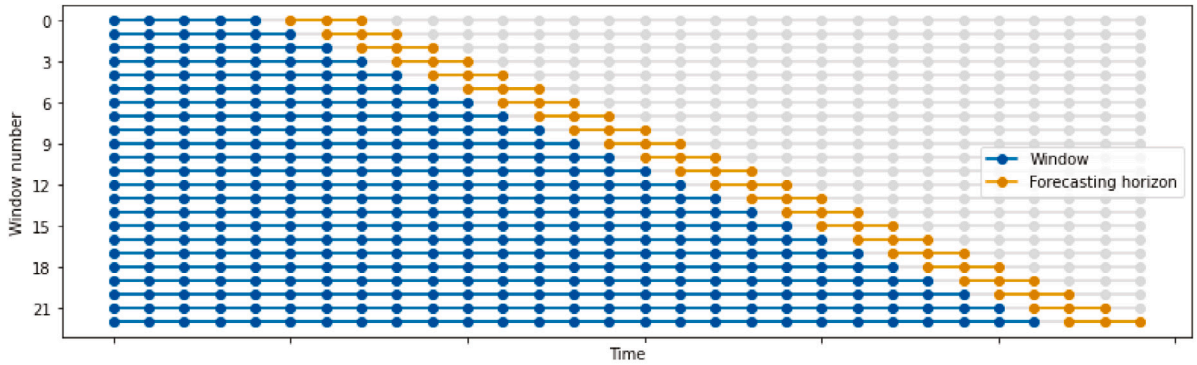



**Fig. 3.** Expanding window example for cross-validation [47].

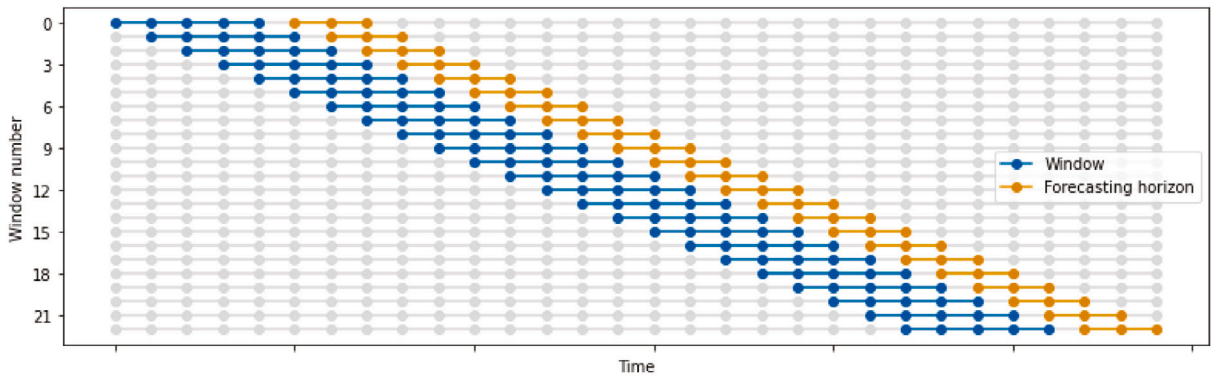



**Fig. 4.** Sliding window example for cross-validation [47].

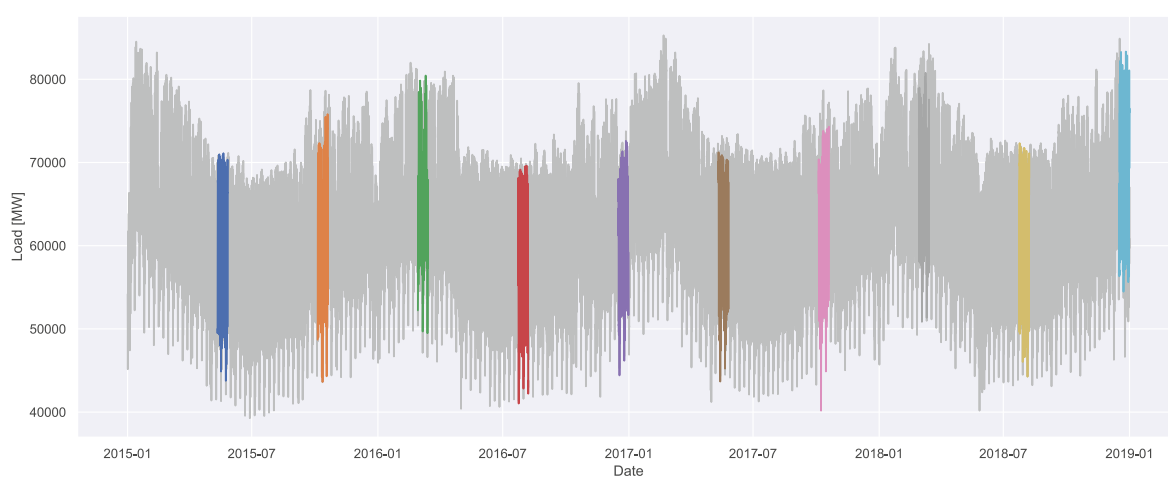
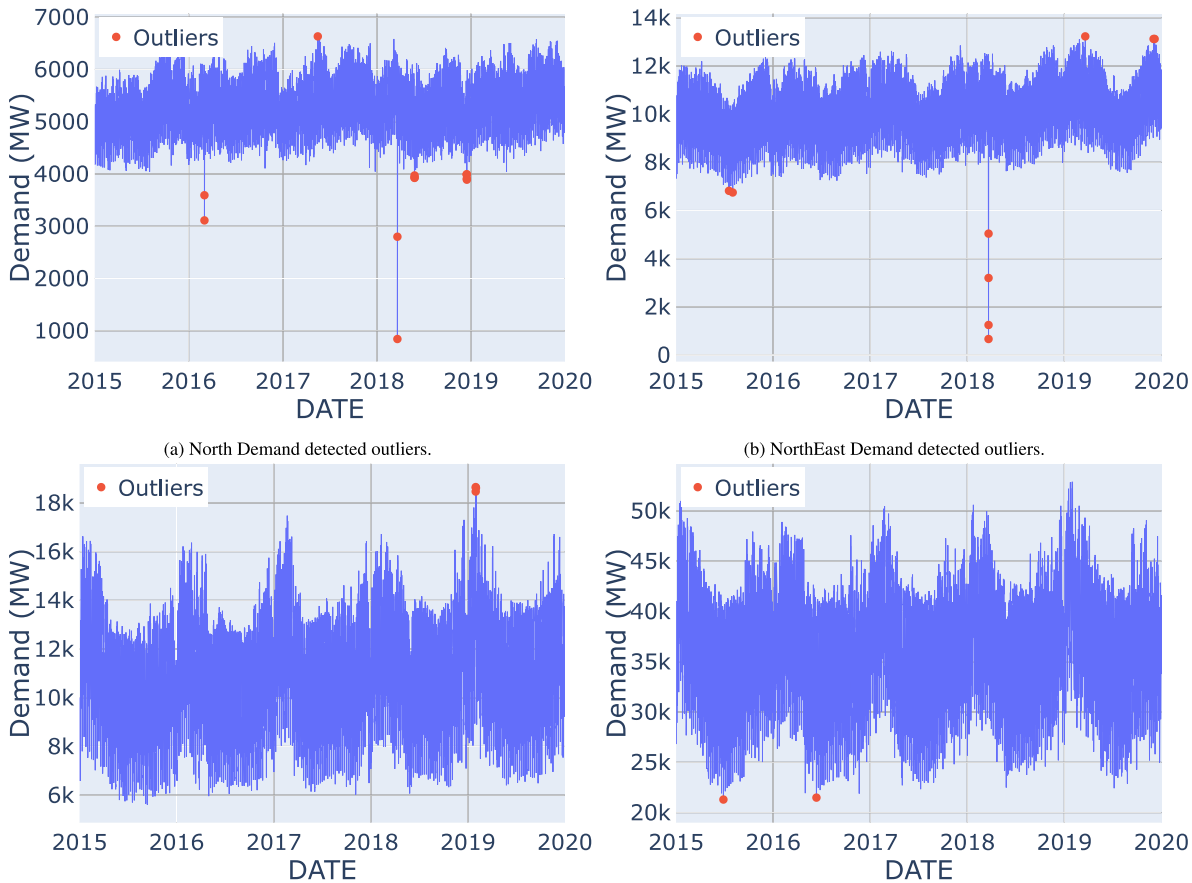



**Fig. 5.** Forecast using 10-fold cross-validation on ONS dataset.

Even if some event could cause this peak or valley, due to intentional shutdown for maintenance or emergency action, this is not helpful for forecasting models since these values are way different compared with the regular measurements and only happen once or twice every three years. Collective outliers can be subsets of novelties in data, such as a signal indicating the discovery of new phenomena. The outliers of ONS dataset shown in Figs. 6(a), 6(b), 6(c), 6(d), for each different region, which has been detected by Local Outlier Factor (LOF) algorithm [50], where it was required to adjust the negative outlier factor to increase its detection performance for both datasets researched.

#### 4.2.2. Interpolation

After detecting and removing outliers, there is a way to replace them properly, for instance, using the interpolation technique. Usually, a linear interpolation is sufficient to fill this gap on the dataset. Nevertheless, it has a better response for point outliers, where only a single or couple of points are outliers in an interval of samples, as shown in Fig. 6(a). Other techniques must be considered and tested to evaluate a better response to the regression model. The outliers, identified as NaN and zero values, were normalized by overwriting them, replacing them with the result of linear interpolating neighbor values.




(a) North Demand detected outliers.

(b) NorthEast Demand detected outliers.

(c) South Demand detected outliers.

(d) SouthEast Demand detected outliers.

**Fig. 6.** ONS dataset analysis.

### 4.3. Feature engineering

Feature engineering is part of the pre-processing stage, which determines the entries or independent variables of the learning model and directly impacts model performance. This work has considered weather data, lagged demand values, and calendar values. To enhance the model training, new data entries have been included in the dataset to have more specific events related to dates, such as holidays and weekdays. Day, month, and year have been readjusted to be on single columns to model process it as a unique feature. There was an inconsistency regarding the hour column, which has been fixed by shifting one hour earlier since the day length was between 1 h to 24 h instead of 0 h to 23 h.

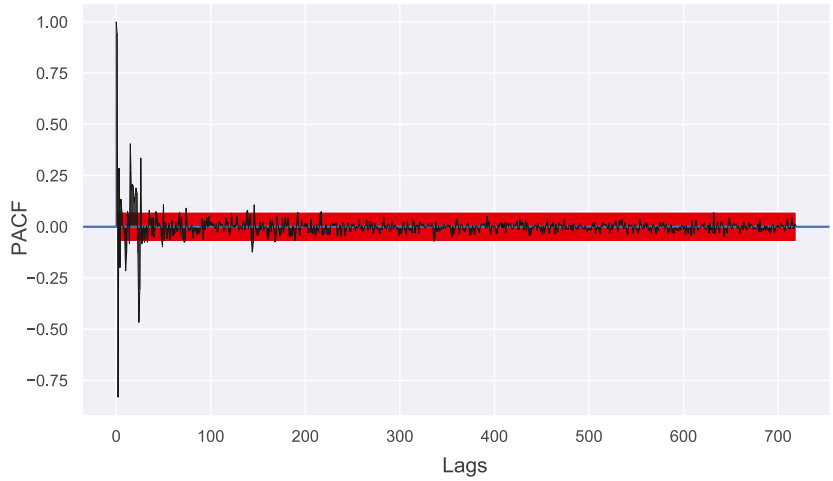
The weather data consists of temperature values or indicates the year's season (e.g., summer, fall, winter, spring), classified as an exogenous variable that can add relevant information for model prediction. As [51] observed, the weather and calendar data are 50% or more significant when compared with other exogenous variables such as socio-economic data (economic trends, GDP, no. of employment), demographic information (birth rate, dwelling count, population) and others (no. of sensors, occupants, devices).

The Partial Autocorrelation Function (PACF) identifies the order of an AR, which can be used to calculate the most significant lags for a time series sequence, as the author's [52] have been done in their respective work. To select the most significant lags of the series, the maximum number of lags must be defined, which could be the total number of samples available in the series. If there is only one sample (one step ahead), then the model would not have any lagged data to train, making forecasting impractical. Thus, to avoid this problem, the maximum number of lags is limited by a quarter of the series, which has been used [53]. To measure the significance of lagged values from time series, the confidence interval of the lag must not be correlated with the actual value. Thereby, its confidence coefficient might be null. A rule often applied in the literature as 95% for confidence interval, considering $1.96\sqrt{N}$, where $N$ is the total number of time series samples. Any lag with correlation coefficient out of confidence interval would be considered significant.

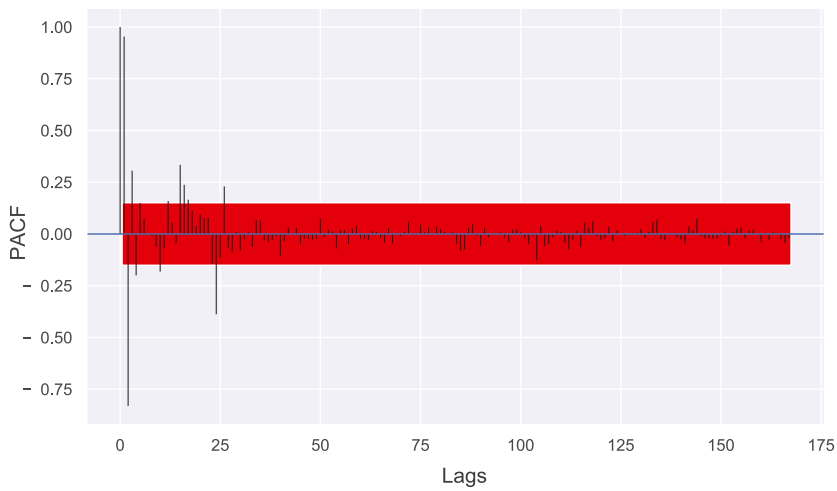
Figs. 7(a) and 7(b) show the PACF for ISO-NE dataset with 700 lags (around 30 days) and 200 lags (around eight days), respectively. The lags are in black, and the confidence interval is in red. Values can notice the significant lags out of the confidence interval, in which most significant terms are before 200 lags. That makes sense since the first values above 0.25 correlation coefficient, as better shown in Fig. 7(b), are the most important ones, followed by some lags at interval 140–170, which marks one week (168 steps-ahead), some patterns start repeating, like weekdays and working-days (e.g., Monday to Monday), as the PACF identify very well the people behavior pattern.

### 4.4. Feature importance

To determine which features are the most important for the model, it is possible to use Random Forest to perform a pre-regression showing which features contribute more to the output, as shown in Fig. 8. The features *Hour*, followed by *DryBulb*, the air temperature, are the most important in the rank defined by the Random Forest algorithm, which is sufficient to perform time series forecasts. The hour of the day defines the people's behavior which will demand more or less power, depending on temperature and seasons of the year, where usually low temperatures demand more power consumption.




(a) Function with 700 lags (around 30 days).




(b) Function with 200 lags (around 8 days).

**Fig. 7.** ISO-NE dataset — Partial autocorrelation.

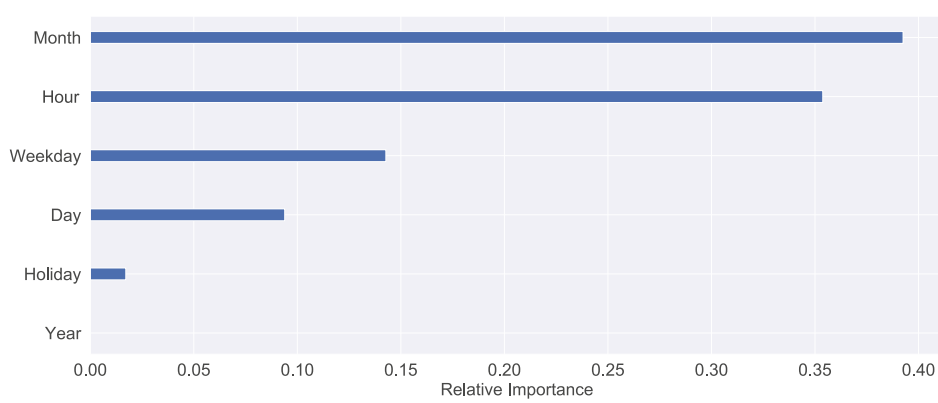



**Fig. 8.** *Feature importance* - Relative importance over each dataset feature presented.

### *4.5. Data transformation*

Data transformation is part of machine learning pre-processing, where some valuable transformations can be made to extract more meaningful inputs for the machine learning model. Examples include calendar adjustments, converting a simple date 2021-28-03 10:00:00 to different variables such as year, month, day, and hour, and mathematical transformations using logarithms, Fourier Series, power transformations, and more. Predictive models take simple patterns more quickly, so the data transformation purpose is to simplify historical patterns by removing known sources of variation or making the pattern more consistent across the entire dataset.

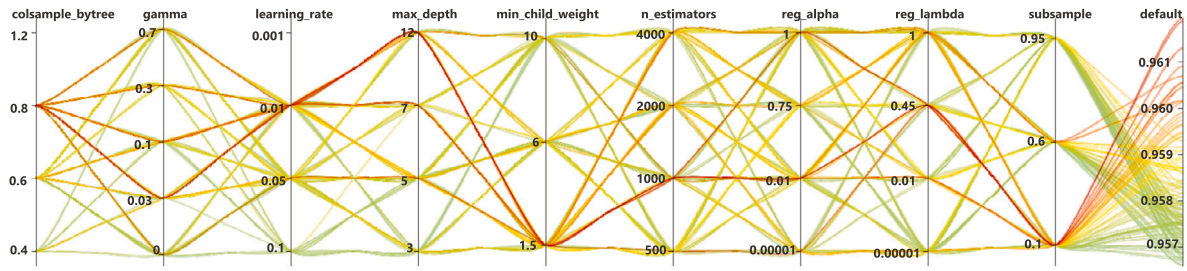



**Fig. 9.** AutoML Neural Network Intelligence (NNI) overview — trial details.

#### 4.5.1. StandardScaler

Standard scaler or center scaling transformation is a way to standardize the observable data by subtracting the arithmetic mean $\mu$ of the training samples and dividing by the standard deviation of the training samples, as shown in Eq. (14). Standardizing datasets is a known requirement for most machine learning estimators, and they might behave unsatisfactorily if this requirement is not fulfilled for the individual features.

As an example, many elements from the objective function of a learning algorithm, such as the Radial Basis Function (RBF) kernel of SVR or the L1 and L2 regularizers of linear models, already assume that all features are centered around zero and have variance in the same order, so if a feature has a variance that its order of magnitude is larger than the others, it might control the objective function and may lead the estimator to be unable to learn from other features correctly as expected. The adopted scaler is given by:

$$Z = \frac{x - \mu}{\sigma}. \tag{14}$$

#### 4.5.2. Box–Cox transformation

Electricity demand time series data are usually pre-processed by Box–Cox transformations to remove any embedded trend and approximate to normal or Gaussian distribution:

$$\omega_t = \begin{cases} log(y_t), & \text{if } \lambda = 0; \\ (y_t^{\lambda} - 1)/\lambda, & \text{otherwise.} \end{cases} \tag{15}$$

where $\lambda$ is the estimated parameter by maximized log-likelihood and $\omega_t$ represents the time series data after Box–Cox transformation and $y_t$ is the original time series data [54]. Power transformations are often defined as $\omega_t = y_t^p$.

#### 4.5.3. Time series dimension transformation

Transforming calendar data from one-dimensional (1D) space to two-dimensional (2D) space may improve the model performance, increasing its accuracy on load forecasting by at least 1.5%; depending on the current model performance circumstances, it may be a wide step on output results. Moon et al. [55] has shown a comparative table where 2D space results explain their correlation more effectively than 1D space results.

### 4.6. Automated Machine Learning (AutoML)

To define the model's best hyperparameters, the AutoML tool has been used in this research. The algorithm runs trial jobs by tuning different structures to search for the best hyperparameters setup. A rapid hyperparameter evaluation can be made by filtering out the top 20% trials, as shown in Fig. 9; thus, the red curves are the best results, followed by yellow and green, which are the worst ones. The best hyperparameters were used to tune the model and to compute the comparisons.

Considering hyperparameter tuning, the Tree-structured Parzen Estimator (TPE) is one of the state-of-the-art algorithms to speed up finding the best hyperparameters for the forecasting models. The TPE is part of the Sequential Model-Based Optimization (SMBO) approach, which sequentially create models to approximate the performance of hyperparameters based on historical measurements.

The TPE is modeled by $p(h|m)$, and $p(m)$, where $h$ represents hyperparameters and $m$ is related to evaluation metrics. TPE models $p(h|m)$ by transforming the generative process replacing the distributions of the configuration prior with non-parametric densities, which is defined by:

$$p(h|m) = \begin{cases} \ell(h), & \text{if } m < m^*, \\ g(h), & \text{if } m \geq m^* \end{cases} \tag{16}$$

where $\ell(m)$ is the density formed by using the observations $h^{(j)}$ such the corresponding loss $f(h^{(j)})$ was less than $m^*$ and $g(h)$ is the density created regarding the remaining observations.

## 5. Results and discussion

The experiments are evaluated considering two datasets ONS and ISO-NE. The training and validation stage consisted in evaluating the mode by the error metrics, such as RMSE, MAE, and MAPE, using the approach with 10-fold cross-validation, with a training stage of 130 days or 3120 steps and validation stage of 15 days-ahead or 360 steps ahead each fold (3600 steps in total forecasted), along the four years of dataset, from 2015 to 2018. The test stage consisted of evaluating the model and forecasting the unforeseen data, which has yet to be used in the training stage.

The test set comprises a training stage of 17 days or 408 steps, and a validation stage of 20 sets of 24 steps ahead, which means the model forecasts one day or 24 steps within 10 different days throughout the 2019 year. Then it was calculated the average and standard deviation of the error metrics RMSE, MAE, and MAPE were used for result evaluation. The period of 2019 for the test set was explicitly chosen to be out of sight of the model training since they are unforeseen data and the tests are consistent with the actual use case.

For the results presented, adding lags as an additional entry for the models was not helping to increase the accuracy, instead was decreasing the models' performance for load forecasting. Somehow, the lags misled the model predictions instead of guiding it to better results. Thus, they were not included in the model.

The computer configuration used to run the experiments has a CPU AMD Ryzen 3600x 3.8 GHz with 32 MB of cash memory, 16 GB of DDR4 random-access memory (RAM), 500 GB of Solid State Drive (SSD), and a Graphics Processing Unit (GPU) GeForce RTX 2060 with 6 GB DDR6 192 bits of internal memory. Considering the k-fold cross-validation method, for $k = 10$ the data was split in 90% for training and 10% for validation, for $k = 20$ the data was split in 95% for training and 5% for validation.

**Table 2**
Computation time for IMFs considering different decomposition methods.

| IMFs | EMD | EEMD | CEEMDAN |
|---|---|---|---|
| | Time (s) | | |
| 1 | 2.5 | 84.0 | 573.0 |
| 2 | 2.9 | 150.7 | 632.1 |
| 3 | 3.0 | 202.7 | 649.7 |
| 4 | 3.2 | 221.1 | 737.8 |
| 5 | 3.2 | 240.2 | 767.8 |
| 6 | 3.2 | 252.6 | 751.9 |
| 7 | 3.1 | 254.7 | 749.4 |
| 8 | 3.1 | 265.5 | 737.8 |
| 9 | 3.5 | 267.9 | 734.7 |

**Table 3**
RMSE for different IMFs for each decomposition method using GBR algorithm for forecasting.

| Number of IMFs | Decomposition methods | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | **3029.1** | 3008.0 | **2961.8** | **3030.0** |
| 2 | | **2916.1** | 3233.2 | 2975.0 | 3156.7 | |
| 3 | **2998.9** | 3022.0 | 3252.3 | 3139.2 | 3236.2 | |
| 4 | | 3021.8 | 3328.1 | 3117.6 | 3319.3 | |
| 5 | | 3005.2 | 3538.9 | 3106.4 | 3216.8 | |
| 6 | | 3006.4 | 3593.3 | **2933.7** | 3413.5 | |
| 7 | | 3037.6 | 3455.1 | 3002.7 | 3378.8 | |
| 8 | | 3031.6 | 3486.6 | 2971.3 | 3356.1 | |
| 9 | | 3018.9 | 3519.8 | 2976.5 | 3357.0 | |

### 5.1. ONS data analysis

This section shows the results of experiments made applying different ML models and decomposition techniques for ONS dataset. The decomposition tests were performed to evaluate the number of IMFs for each decomposition method, and how the RMSE and the computation time are affected due to different configurations of IMF quantity.

In the following subsections, both tests for computation complexity and RMSE are shown in detail. Considering the computation time with nine IMFs to decompose each dataset, ONS and ISO-NE, from 2015 to 2018 years range, STL-A has the shortest duration at 0.027 s, followed by EWT at 0.479 s. EMD takes 3.044 s and EEMD takes 247.368 s. CEEMDAN takes the longest at 713.165 s. Although these high computation times do not affect the model tuning, either model training or test since the decomposed values are saved previously and then used during the forecasting execution. However, they affect when the tuning of decomposition methods is done.

Table 2 shows the computation time for the used decomposition methods. The EMD times look pretty much the same for all modes. The EEMD times grow accordingly with IMF number, and the CEEMDAN times grow until IMF equals to five, then the following timings are similar or shorter. To evaluate the best number of IMFs for each decomposition method, it was performed test measuring the RMSE from one to nine IMFs as shown in Table 3. The Seasonal and Trend decomposition using Locally Estimated Scatterplot Smoothing (STL) decomposition has no configuration to change the number of IMFs since it is fixed to decompose to trend, seasonal, and residue components. The EWT decomposition cannot be executed with one IMF.

In Table 3 is possible to see clearly which IMF is the best for each decomposition method, ranking by the lowest RMSE metric. The STL is only for RMSE reference. For GBR algorithm, the best IMFs modes are two modes for EWT; 1 mode for EMD; 6 modes for EEMD; 1 mode for CEEMDAN. EEMD and EWT have the lowest RMSE among the decomposition methods tested, which is not relevant right now since no hyperparameters tuning were made. However, they are already lower than the none decomposition method.

**Table 4**
RMSE for different IMFs for each decomposition method using XGBoost algorithm for forecasting.

| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | **2892.4** | 2930.2 | **2968.8** | **2922.9** |
| 2 | | 2854.6 | 2938.9 | 2967.1 | 2972.7 | |
| 3 | **2918.8** | 3024.0 | 3298.5 | 2962.7 | 3377.4 | |
| 4 | | 2949.9 | 3691.0 | 3036.1 | 3618.0 | |
| 5 | | 2978.7 | 3999.9 | **2882.0** | 3632.3 | |
| 6 | | 2950.9 | 3920.9 | 2970.6 | 3546.6 | |
| 7 | | 3099.7 | 3666.0 | 2991.5 | 3546.0 | |
| 8 | | 3112.5 | 3679.3 | 2883.4 | 3560.7 | |
| 9 | | **2162.8** | 3690.8 | 2903.4 | 3554.6 | |

**Table 5**
RMSE for different IMFs for each decomposition method using SVR algorithm for forecasting.

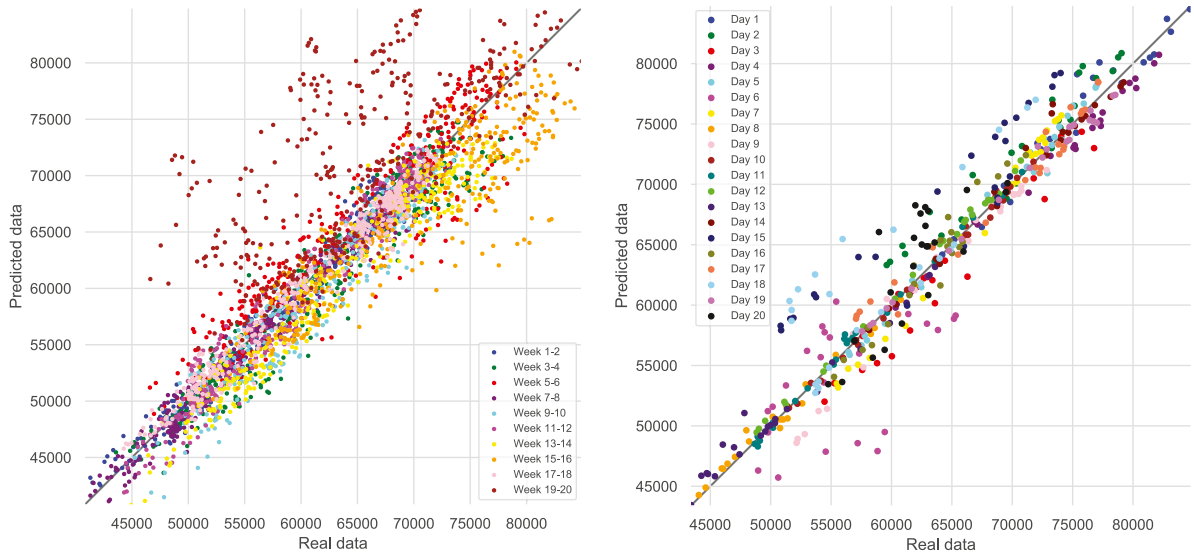
| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | **4368.9** | **4469.3** | **4374.6** | **4305.8** |
| 2 | | 5123.7 | 4777.6 | 4476.2 | 4751.3 | |
| 3 | **5280.4** | 4801.7 | 5113.7 | 5217.2 | 5154.7 | |
| 4 | | 4821.5 | 5612.5 | 5138.8 | 5654.3 | |
| 5 | | **4684.2** | 5591.9 | 5704.7 | 6134.8 | |
| 6 | | 4798.7 | 5828.8 | 5881.3 | 6540.6 | |
| 7 | | 4934.8 | 5925.8 | 6452.9 | 6526.3 | |
| 8 | | 5166.7 | 6087.2 | 6485.1 | 6613.9 | |
| 9 | | 4894.2 | 6120.6 | 6513.4 | 6633.0 | |

**Table 6**
RMSE for different IMFs for each decomposition method using kNN algorithm for forecasting.

| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | **4670.2** | **4672.6** | **4670.2** | **4670.2** |
| 2 | | 4745.8 | 4670.2 | 4672.6 | 4670.2 | |
| 3 | **4670.2** | 4722.9 | 4670.2 | 4672.6 | 4670.2 | |
| 4 | | 4675.4 | 4670.2 | 4672.6 | 4670.2 | |
| 5 | | 4671.8 | 4670.2 | 4672.6 | 4670.2 | |
| 6 | | **4669.2** | 4670.2 | 4672.6 | 4670.2 | |
| 7 | | 4669.2 | 4670.2 | 4672.6 | 4670.2 | |
| 8 | | 4673.5 | 4670.2 | 4672.6 | 4670.2 | |
| 9 | | 4677.9 | 4670.2 | 4672.6 | 4670.2 | |

Table 4 presents the best IMF for XGBoost algorithm of each decomposition method, ranking by the lowest RMSE metric and the best IMFs that are 9 modes for EWT, 1 mode for EMD, 5 modes for EEMD, and 1 mode for CEEMDAN. The EWT and EEMD have the lowest RMSE among the decomposition methods tested, and they are lower than the none decomposition method. Table 5 shows the best IMF for SVR algorithm of each decomposition method, ranking by the lowest RMSE metric, and the best IMFs modes are 5 modes for EWT; 1 mode for EMD; 1 mode for EEMD; 1 mode for CEEMDAN. EMD and CEEMDAN have the lowest RMSE among the decomposition methods tested, and the none decomposition method has the lowest error. Table 6 shows the best IMF for kNN algorithm of each decomposition method, ranking by the lowest RMSE metric, and the best IMFs modes are 6 modes for EWT; 1 mode for EMD; 1 mode for EEMD; 1 mode for CEEMDAN. EWT has the lowest RMSE among the decomposition methods tested, and it is lower than the none decomposition method. However, all decomposition methods have very similar results.

The bias test was made to verify how far the model was forecasting the data for training and test sets, as shown in Figs. 10(a) and 10(b). The values over-forecasted by the model are sitting above the cross-line, and the values under-forecasted are below the cross-line. For the training set, there are some over-forecasted values for one fold of the




(a) (b)

**Fig. 10.** ONS dataset bias verification (a) of the training set where each different color of scattered data represents the considered time (given a two weeks time window), (b) of the test set where each different color of scattered data represents 1 of 20 days selected in 2019.

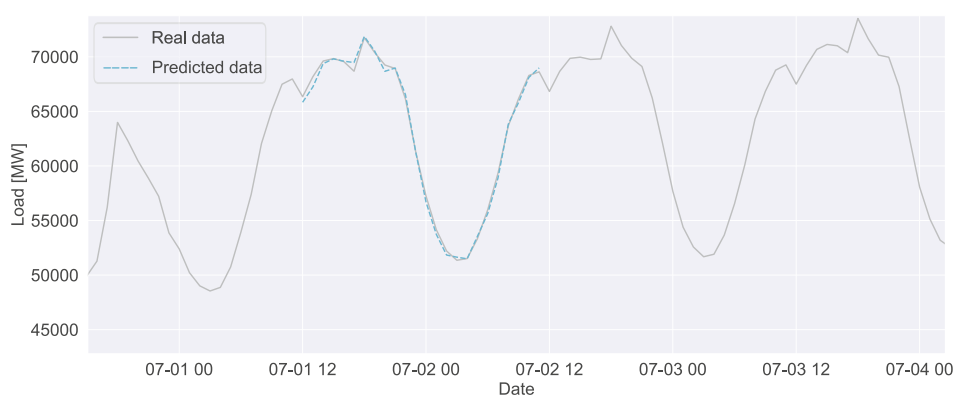



**Fig. 11.** ONS dataset — One sample of the forecasted set of best test set model (XGBoost + EWT with nine modes, no tuning.).

training set, which can be observed as light blue circles. Few over-forecasted values exist for the test set, but less than the training set. Overall, the forecasted values are spread evenly over the cross-line reference. The test was performed with the best model (with lowest RMSE) observed on test set results, in this case, EWT with 9 IMF modes using XGBoost algorithm, without hyperparameters tuning.

The overall results presented for the training set, shown in Table 7, the EWT decomposition method using XGBoost algorithm performed better than other methods, for tuned and not tuned hyperparameters. Although the results for the test set, shown in Table 8, the EWT decomposition using XGBoost algorithm performed better among all tests. For tuned hyperparameters, the EEMD decomposition using XGBoost algorithm had the lowest RMSE. The total time of model training was around 517.4 h. The total execution time of all tests, including training and test set, was around 1.80 h.

The best result for training set ONS dataset was EWT decomposition method with XGBoost, RMSE with 2477.8 MW, MAE with 1925.3 MW and MAPE with 3.08%, configured to nine IMFs and with model tuning. The best result for the test set of ONS dataset was EWT with XGBoost, RMSE with 1931.8 MW, MAE with 1564.9 MW and MAPE with 2.54%, configured to nine IMFs without any hyperparameters tuning.

Then, the best model for the test set was not the hyperparameters tuned but the one without tuning (Fig. 11). Even using stacking ensembles, XGBoost + GBR, XGBoost + GBR + SVR + kNN and so on, the results were still far from expected. The reason behind that might be related to cross-validation setup, 15 days for the validation set with 10-fold, in total 360 steps ahead by fold or 3600 steps ahead to forecast, was not sufficient for hyperparameters tuning, letting too many empty spaces along the period of 2015 to 2018. Maybe using 20-fold, with 75 days for the training stage and 15 days ahead for the validation stage, was already an adjustment needed for producing better results.

These results were not tuned; however, the models performed better, probably due to an overfitting problem. The model was trained too much on the same range and data sequence with few disturbances and differences along the training set. Alternatively, even the amount of data needed to be increased for model robustness produces results incompatible with the expected ones for the test set.

### 5.2. ISO-NE Data Analysis

This section shows the results from different ML models and decomposition techniques applied in this dataset for ISO-NE utility. ML models used were GBR, XGBoost, kNN and SVR.

**Table 7**
ONS dataset — Training set results from 2015 to 2018 period, with 10-fold cross-validation, 360 steps ahead (15 days ahead) each fold.

| Algorithm | Method | IMFs | Tuning | RMSE (MW) | MAE (MW) | MAPE (%) | Duration (s) | Tuning time (h) |
|---|---|---|---|---|---|---|---|---|
| GBR | CEEMDAN | 1 | No | 3410.8 (± 1634.7) | 2728.6 (± 1270.4) | 4.39 (± 2.04) | 20 | **0**[a] |
| GBR | CEEMDAN | 1 | Yes | 3314.9 (± 1728.5) | 2653.5 (± 1287.3) | 4.30 (± 2.10) | 84 | 26.6 |
| GBR | EEMD | 6 | No | 2935.8 (± 1618.8) | 2305.8 (± 1294.0) | 3.70 (± 2.04) | 20 | **0**[a] |
| GBR | EEMD | 6 | Yes | 3174.1 (± 1487.1) | 2534.5 (± 1173.1) | 4.06 (± 1.82) | 447 | 69.3 |
| GBR | EMD | 1 | No | 3029.1 (± 1576.2) | 2352.7 (± 1198.6) | 3.78 (± 1.89) | 15 | **0**[a] |
| GBR | EMD | 1 | Yes | 2975.7 (± 1644.5) | 2337.4 (± 1280.1) | 3.71 (± 1.94) | 1862 | 105.5 |
| GBR | EWT | 2 | No | 2916.1 (± 1761.3) | 2271.3 (± 1354.2) | 3.64 (± 2.18) | 14 | **0**[a] |
| GBR | EWT | 2 | Yes | 2608.5 (± 1566.9) | 2105.7 (± 1274.0) | 3.36 (± 1.97) | 342 | 22.3 |
| GBR | None | – | No | 3030.0 (± 1543.7) | 2374.8 (± 1163.0) | 3.81 (± 1.82) | 12 | **0**[a] |
| GBR | STL-A | – | No | 2998.9 (± 1650.8) | 2307.5 (± 1248.9) | 3.70 (± 2.00) | 14 | **0**[a] |
| GBR | STL-A | – | Yes | 2755.5 (± 1570.0) | 2203.9 (± 1240.0) | 3.54 (± 1.96) | 170 | 31.3 |
| KNN | None | – | No | 4670.2 (± 1438.1) | 3707.5 (± 1221.5) | 6.09 (± 1.93) | **4** | **0**[a] |
| SVR | EMD | 1 | No | 4368.9 (± 1041.3) | 3564.8 (± 1041.7) | 5.80 (± 1.72) | 40 | **0**[a] |
| SVR | None | – | No | 4305.8 (± 1015.6) | 3493.1 (± 1016.2) | 5.66 (± 1.65) | 41 | **0**[a] |
| XGBoost | CEEMDAN | 1 | No | 2968.8 (± 1808.9) | 2315.7 (± 1365.1) | 3.69 (± 2.14) | 12 | **0**[a] |
| XGBoost | CEEMDAN | 1 | Yes | 2743.1 (± 1571.5) | 2146.3 (± 1218.2) | 3.39 (± 1.88) | 74 | 15.8 |
| XGBoost | EEMD | 5 | No | 2882.0 (± 1546.7) | 2271.7 (± 1256.6) | 3.62 (± 1.91) | 18 | **0**[a] |
| XGBoost | EEMD | 5 | Yes | 2883.5 (± 1517.5) | 2272.3 (± 1238.0) | 3.62 (± 1.89) | 45 | 19.1 |
| XGBoost | EMD | 5 | No | 2892.4 (± 1766.9) | 2262.3 (± 1381.6) | 3.61 (± 2.15) | 14 | **0**[a] |
| XGBoost | EMD | 5 | Yes | 2682.0 (± 1649.3) | 2107.1 (± 1276.8) | 3.34 (± 1.99) | 38 | 81.7 |
| XGBoost | EWT | 9 | No | 3081.1 (± 2162.8) | 2397.8 (± 1616.2) | 3.83 (± 2.58) | 746 | **0**[a] |
| XGBoost | EWT | 9 | Yes | **2477.8 (± 1771.8)** | **1925.3 (± 1296.8)** | **3.08 (± 2.05)** | 1214 | 109.9 |
| XGBoost | None | – | No | 2922.9 (± 1845.0) | 2247.9 (± 1377.2) | 3.59 (± 2.16) | 12 | **0**[a] |
| XGBoost | STL-A | – | No | 2918.8 (± 1612.6) | 2319.1 (± 1293.5) | 3.70 (± 1.99) | 13 | **0**[a] |
| XGBoost | STL-A | – | Yes | 2704.9 (± 1663.0) | 2107.2 (± 1258.0) | 3.34 (± 1.96) | 33 | 35.9 |
| XGB+SVR | None | – | No | 3004.3 (± 1780.2) | 2338.3 (± 1301.2) | 3.78 (± 2.05) | 218 | **0**[a] |
| XGB+GBR+SVR+KNN | None | – | No | 2920.7 (± 1638.1) | 2282.1 (± 1237.2) | 3.69 (± 1.96) | 218 | **0**[a] |
| XGB+KNN | None | – | No | 2878.9 (± 1800.4) | 2240.9 (± 1350.9) | 3.59 (± 2.12) | 251 | **0**[a] |
| XGB+GBR | None | – | No | 2746.3 (± 1732.4) | 2155.0 (± 1291.9) | 3.47 (± 2.04) | 241 | **0**[a] |
| | | | | | | **Total** | 6228 | 517.4 |

[a] Tuning time (h) with value zero means less than one hour.

**Table 8**
ONS dataset — Test set results for the 2019 period, with 20 sets of 24 steps ahead (1 day ahead).

| Algorithm | Method | IMFs | Tuning | RMSE (MW) | MAE (MW) | MAPE (%) | Duration (s) |
|---|---|---|---|---|---|---|---|
| GBR | CEEMDAN | 1 | No | 2627.7 (± 1229.5) | 2182.1 (± 1078.7) | 3.56 (± 2.02) | 3 |
| GBR | CEEMDAN | 1 | Yes | 2748.4 (± 1244.1) | 2316.5 (± 1047.0) | 3.77 (± 1.95) | 45 |
| GBR | EEMD | 6 | No | 2255.7 (± 1304.2) | 1936.1 (± 1186.3) | 3.15 (± 2.04) | 3 |
| GBR | EEMD | 6 | Yes | 2654.8 (± 1275.8) | 2258.8 (± 1048.4) | 3.72 (± 2.00) | 197 |
| GBR | EMD | 1 | No | 2251.2 (± 1216.3) | 1905.4 (± 1055.8) | 3.07 (± 1.78) | 1 |
| GBR | EMD | 1 | Yes | 2584.4 (± 1568.2) | 2219.8 (± 1400.3) | 3.54 (± 2.23) | 148 |
| GBR | EWT | **2** | No | 2572.3 (± 1756.5) | 2222.3 (± 1557.6) | 3.56 (± 2.53) | 1 |
| GBR | EWT | 2 | Yes | 2861.1 (± 1835.2) | 2480.8 (± 1632.3) | 3.96 (± 2.66) | 236 |
| GBR | None | – | No | 2241.2 (± 1242.7) | 1886.8 (± 1072.1) | 3.07 (± 1.85) | 1 |
| GBR | STL-A | – | No | 2319.3 (± 1681.4) | 1965.1 (± 1443.6) | 3.19 (± 2.49) | 2 |
| GBR | STL-A | – | Yes | 2511.0 (± 1599.9) | 2144.2 (± 1372.4) | 3.55 (± 2.46) | 78 |
| KNN | None | – | No | 4679.7 (± 2580.8) | 3990.5 (± 2291.6) | 6.65 (± 4.28) | **0**[a] |
| SVR | EMD | 1 | No | 5359.5 (± 2942.2) | 4518.3 (± 2718.9) | 7.32 (± 4.72) | 1 |
| SVR | None | – | No | 5838.5 (± 4906.5) | 4767.9 (± 3882.2) | 7.75 (± 6.85) | 1 |
| XGBoost | CEEMDAN | 1 | No | 2344.0 (± 1524.2) | 1966.0 (± 1333.4) | 3.16 (± 2.14) | 1 |
| XGBoost | CEEMDAN | 1 | Yes | 2588.2 (± 1284.1) | 2219.8 (± 1124.4) | 3.63 (± 1.91) | 24 |
| XGBoost | EEMD | 5 | No | 2443.8 (± 1396.7) | 2053.6 (± 1203.9) | 3.31 (± 1.98) | 3 |
| XGBoost | EEMD | 5 | Yes | 2510.3 (± 1303.0) | 2111.9 (± 1128.0) | 3.41 (± 1.86) | 17 |
| XGBoost | EMD | 5 | No | 2333.2 (± 1552.0) | 1948.4 (± 1344.9) | 3.12 (± 2.17) | 1 |
| XGBoost | EMD | 5 | Yes | 2601.7 (± 1576.4) | 2174.9 (± 1244.0) | 3.59 (± 2.25) | 13 |
| XGBoost | EWT | 9 | No | **1931.8 (± 1511.3)** | **1564.9 (± 1246.9)** | **2.54 (± 2.14)** | 55 |
| XGBoost | EWT | 9 | Yes | 2885.4 (± 1940.2) | 2543.2 (± 1664.0) | 4.29 (± 3.15) | 345 |
| XGBoost | None | – | No | 2308.9 (± 1588.5) | 1954.8 (± 1405.9) | 3.13 (± 2.26) | 1 |
| XGBoost | STL-A | – | No | 2193.4 (± 1697.9) | 1899.9 (± 1520.9) | 3.14 (± 2.65) | 1 |
| XGBoost | STL-A | – | Yes | 2603.4 (± 1335.4) | 2321.7 (± 1202.5) | 3.85 (± 2.31) | 12 |
| XGB+SVR | None | – | No | 2422.7 (± 1440.6) | 2064.7 (± 1266.1) | 3.25 (± 1.96) | 9 |
| XGB+GBR+SVR+KNN | None | – | No | 2363.7 (± 1310.8) | 1981.3 (± 1108.3) | 3.16 (± 1.83) | 12 |
| XGB+KNN | None | – | No | 2476.5 (± 1462.9) | 2135.7 (± 1314.1) | 3.35 (± 2.01) | 9 |
| XGB+GBR | None | – | No | 2283.0 (± 1306.4) | 1910.1 (± 1129.3) | 3.05 (± 1.86) | 12 |
| | | | | | | **Total** | 1234 |

[a] Duration (s) with value zero means less than one second.

**Table 9**
RMSE for different IMFs for each decomposition method using GBR algorithm for forecasting.

| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | **1399.1** | 1442.2 | **1412.9** | **1375.0** |
| 2 | | **1500.8** | 1509.8 | 1433.5 | 1527.4 | |
| 3 | **1520.8** | 1552.6 | 1525.3 | **1409.6** | 1546.0 | |
| 4 | | 1508.7 | 1468.0 | 1484.8 | 1588.5 | |
| 5 | | 1499.3 | 1518.5 | 1497.5 | 1527.1 | |
| 6 | | 1545.4 | 1490.7 | 1465.2 | 1512.6 | |
| 7 | | 1579.2 | 1569.2 | 1507.4 | 1519.3 | |
| 8 | | 1518.9 | 1539.0 | 1539.1 | 1543.8 | |
| 9 | | 1536.2 | 1525.0 | 1536.9 | 1487.2 | |

**Table 10**
RMSE for different IMFs for each decomposition method using XGBoost algorithm for forecasting.

| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | 1726.2 | 1662.2 | 1644.8 | **1643.3** |
| 2 | | 1740.0 | 1931.0 | 1642.2 | 1735.0 | |
| 3 | **1647.9** | 1666.3 | 1885.4 | 1625.7 | 1737.6 | |
| 4 | | 1541.3 | 1826.4 | 1663.7 | 1741.6 | |
| 5 | | **1506.4** | 1724.4 | 1647.2 | 1633.7 | |
| 6 | | 1552.7 | 1737.7 | 1615.7 | 1627.8 | |
| 7 | | 1520.9 | 1747.7 | 1541.0 | 1651.2 | |
| 8 | | 1505.8 | 1688.1 | 1572.5 | **1605.7** | |
| 9 | | 1510.4 | **1683.6** | **1512.3** | 1611.2 | |

**Table 11**
RMSE for different IMFs for each decomposition method using kNN algorithm for forecasting.

| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | **2095.0** | **2095.9** | **2095.0** | **2095.0** |
| 2 | | 2136.2 | 2095.0 | 2095.9 | 2095.0 | |
| 3 | **2095.0** | 2112.6 | 2095.0 | 2095.9 | 2095.0 | |
| 4 | | **2100.3** | 2095.0 | 2095.9 | 2095.0 | |
| 5 | | 2101.0 | 2095.0 | 2095.9 | 2095.0 | |
| 6 | | 2100.8 | 2095.0 | 2095.9 | 2095.0 | |
| 7 | | 2104.9 | 2095.0 | 2095.9 | 2095.0 | |
| 8 | | 2110.5 | 2095.0 | 2095.9 | 2095.0 | |
| 9 | | 2114.0 | 2095.0 | 2095.9 | 2095.0 | |

**Table 12**
RMSE for different IMFs for each decomposition method using SVR algorithm for forecasting.

| Number of IMFs | Decomposition method | | | | | |
|---|---|---|---|---|---|---|
| | STL-A | EWT | EMD | EEMD | CEEMDAN | None |
| 1 | | | 2807.0 | 2831.8 | 2816.4 | **2833.1** |
| 2 | | **2948.3** | **2783.9** | 2834.9 | **2791.2** | |
| 3 | **3041.3** | 2960.6 | 2800.3 | **2807.0** | 2795.0 | |
| 4 | | 2964.9 | 2803.1 | 2824.4 | 2793.2 | |
| 5 | | 2964.1 | 2807.6 | 2812.6 | 2807.8 | |
| 6 | | 2962.7 | 2832.9 | 2817.7 | 2834.1 | |
| 7 | | 2964.4 | 2825.8 | 2849.0 | 2840.4 | |
| 8 | | 2964.0 | 2838.1 | 2861.6 | 2844.7 | |
| 9 | | 2966.2 | 2860.7 | 2852.0 | 2814.3 | |

The decomposition tests were also performed for ISO-NE dataset, to determine the best configuration of IMF quantity for each decomposition method and algorithm, which were based on RMSE metric. To evaluate the best number of IMFs for each decomposition method, it was performed test measuring the RMSE from one to nine IMFs as shown in Table 9. The STL decomposition has no configuration to change the number of IMFs since it is fixed to decompose to trend, seasonal, and residue components. The same is true for None, which has no decomposition method applied. Moreover, the EWT decomposition cannot be executed with one IMF.

Table 9 shows the best IMF for GBR algorithm of each decomposition method, ranking by the lowest RMSE metric and the best IMFs modes. They are 2 modes for EWT, 1 mode for EMD, 3 modes for EEMD, and 1 mode for CEEMDAN. EMD and EEMD have the lowest RMSE among the decomposition methods tested, but none decomposition has the lowest error. Table 10 shows the best IMF for XGBoost algorithm of each decomposition method, ranking by the lowest RMSE metric and the best IMFs modes. The selected modes are 5 modes for EWT, 9 modes for EMD, 9 modes for EEMD, and 8 modes for CEEMDAN. The EWT and EEMD have the lowest RMSE among the decomposition methods tested, and they are lower than the none decomposition method.

Table 11 shows the best IMF for kNN algorithm of each decomposition method, ranking by the lowest RMSE metric and the best IMFs modes, where 4 modes for EWT, 1 mode for EMD, 1 mode for EEMD, and 1 mode for CEEMDAN were assumed. All decomposition methods have very similar results. Table 12 shows the best IMF for SVR algorithm of each decomposition method, ranking by the lowest RMSE metric and the best IMFs modes. In this case, 2 modes for EWT, 2 modes for EMD, 3 modes for EEMD, and 2 modes for CEEMDAN were considered. As it can be seen, EMD and CEEMDAN have the lowest RMSE among the decomposition methods tested, and they are lower than the none decomposition method.

The overall results presented for the training set, shown in Table 13, the EEMD decomposition method using XGBoost performed better than others models, based on RMSE, for not tuned hyperparameters. However, the lowest MAE and MAPE, the XGBoost without any decomposition, got better results. Although the results for the test set, shown in Table 14, the EWT decomposition using XGBoost performed better among all tests. The GBR without any decomposition had the lowest RMSE, MAE, and MAPE. The total execution time of all tests, including training and test set, was around 13.1 min. The best result for a training set of ISO-NE dataset was EEMD with XGBoost, RMSE with 848.85 MW, configured to nine IMFs without hyperparameters tuning. The XGBoost without any decomposition with MAE with 736.10 MW and MAPE with 5.36% without hyperparameters tuning. The best result for the test set of ISO-NE dataset was no decomposition method with GBR, RMSE with 1375.0 MW, MAE with 1042.7 MW and MAPE with 7.38%, without any hyperparameters tuning.

The bias test was made to verify how far way the model was forecasting the data for training and test sets, as shown in Figs. 12(a) and 12(b). As previously explained, the values over-forecasted by the model are above the cross-line, and the values under-forecasted are below the cross-line. For the training set, the forecasted values are widespread, as observed in Fig. 12(a), noticing that the model has a hard time predicting precisely. However, no bias was observed. There are a few under-forecasted values below the cross-line reference for the test set, observed as a blue circle in Fig. 12(b), so some bias for this specific set was observed. Besides that, the overall forecasted values are spread over the cross-line reference. The test was performed with the best model (with lowest RMSE) observed on test set results, in this case, None decomposition method using GBR algorithm, without hyperparameters tuning.

## 6. Conclusion

Load forecasting using ML models combined with decomposition techniques may be an alternative way to improve model accuracy since the former forecasting method may have reached its limits. Even with new learning algorithms, some intrinsic components of the time series may be embedded, and extracting these components is the key to the next steps in load forecasting. Signal decomposition opens new avenues

**Table 13**
ISO-NE dataset — Training set results for the 2015–2018 period, with 20-fold cross-validation, 360 steps ahead (15 days ahead) each fold.

| Algorithm | Decomposition | IMFs | Tuning | RMSE (MW) | MAE (MW) | MAPE (%) | Duration (s) |
|---|---|---|---|---|---|---|---|
| GBR | CEEMDAN | 1 | No | 932.23 (± 695.39) | 816.07 (± 675.05) | 5.89 (± 3.75) | 50 |
| GBR | EEMD | 3 | No | 952.70 (± 753.12) | 829.10 (± 734.14) | 5.97 (± 4.14) | 24 |
| GBR | EMD | 1 | No | 940.60 (± 682.84) | 821.30 (± 671.46) | 5.93 (± 3.68) | 1 |
| GBR | EWT | 2 | No | 921.57 (± 697.20) | 797.20 (± 680.13) | 5.82 (± 4.03) | 1 |
| GBR | None | – | No | 909.54 (± 662.46) | 800.30 (± 654.21) | 5.79 (± 3.67) | 1 |
| GBR | STL-A | – | No | 921.83 (± 761.04) | 792.70 (± 738.59) | 5.76 (± 4.28) | 1 |
| *k*NN | CEEMDAN | 1 | No | 1215.2 (± 789.63) | 1045.9 (± 762.99) | 7.77 (± 4.87) | 49 |
| *k*NN | EEMD | 1 | No | 1209.7 (± 786.92) | 1038.6 (± 761.77) | 7.71 (± 4.85) | 9 |
| *k*NN | EMD | 1 | No | 1215.2 (± 789.63) | 1045.9 (± 762.99) | 7.77 (± 4.87) | **0**[a] |
| *k*NN | EWT | 4 | No | 1267.2 (± 819.54) | 1082.1 (± 791.67) | 8.08 (± 5.20) | **0**[a] |
| *k*NN | None | – | No | 1215.2 (± 789.63) | 1045.9 (± 762.99) | 7.77 (± 4.87) | **0**[a] |
| *k*NN | STL-A | – | No | 1215.2 (± 789.63) | 1045.9 (± 762.99) | 7.77 (± 4.87) | **0**[a] |
| SVR | CEEMDAN | 2 | No | 2107.5 (± 896.57) | 1819.9 (± 839.33) | 14.3 (± 6.18) | 51 |
| SVR | EEMD | 3 | No | 2100.5 (± 865.46) | 1810.5 (± 778.17) | 14.2 (± 5.21) | 22 |
| SVR | EMD | 2 | No | 2102.3 (± 913.05) | 1821.5 (± 844.71) | 14.2 (± 6.00) | 1 |
| SVR | EWT | 2 | No | 2158.2 (± 861.28) | 1824.0 (± 804.96) | 14.6 (± 5.79) | **0**[a] |
| SVR | None | – | No | 2128.9 (± 800.70) | 1798.0 (± 745.11) | 14.4 (± 5.10) | **0**[a] |
| SVR | STL-A | – | No | 2219.7 (± 830.03) | 1848.9 (± 823.55) | 15.1 (± 6.47) | **0**[a] |
| XGBoost | CEEMDAN | 8 | No | 973.19 (± 681.51) | 830.39 (± 611.81) | 6.21 (± 3.85) | 112 |
| XGBoost | EEMD | 9 | No | **848.85 (± 687.57)** | 743.41 (± 668.53) | 5.37 (± 3.69) | 32 |
| XGBoost | EMD | 9 | No | 974.08 (± 676.06) | 845.62 (± 616.73) | 6.27 (± 3.97) | 3 |
| XGBoost | EWT | 5 | No | 953.72 (± 689.29) | 808.66 (± 650.02) | 6.03 (± 3.93) | 2 |
| XGBoost | None | – | No | 851.55 (± 694.01) | **736.10 (± 656.25)** | **5.36 (± 3.94)** | 1 |
| XGBoost | STL-A | – | No | 892.79 (± 798.27) | 781.42 (± 775.56) | 5.70 (± 4.68) | 1 |
| | | | | | | **Total** | 363 |

[a] Duration (s) with value zero means less than one second.

**Table 14**
ISO-NE dataset — Test set results for the 2019 period, with 20 sets of 24 steps ahead (1 day ahead).

| Algorithm | Decomposition | IMFs | Tuning | RMSE (MW) | MAE (MW) | MAPE (%) | Duration (s) |
|---|---|---|---|---|---|---|---|
| GBR | CEEMDAN | 1 | No | 1412.9 (± 414.91) | 1089.8 (± 358.84) | 7.69 (± 2.07) | 52 |
| GBR | EEMD | 3 | No | 1409.6 (± 428.64) | 1088.4 (± 360.02) | 7.76 (± 2.24) | 27 |
| GBR | EMD | 1 | No | 1399.1 (± 426.75) | 1084.0 (± 376.45) | 7.70 (± 2.36) | 3 |
| GBR | EWT | 2 | No | 1500.8 (± 369.28) | 1186.7 (± 311.75) | 8.48 (± 1.93) | 4 |
| GBR | None | – | No | **1375.0 (± 407.86)** | **1042.7 (± 355.13)** | **7.38 (± 2.22)** | 3 |
| GBR | STL-A | – | No | 1520.8 (± 486.34) | 1191.3 (± 456.28) | 8.49 (± 3.01) | 4 |
| *k*NN | CEEMDAN | 1 | No | 2095.0 (± 912.16) | 1653.3 (± 804.72) | 11.7 (± 5.35) | 5 |
| *k*NN | EEMD | 1 | No | 2095.9 (± 918.92) | 1653.7 (± 808.60) | 11.7 (± 5.37) | 4 |
| *k*NN | EMD | 1 | No | 2095.0 (± 912.16) | 1653.3 (± 804.72) | 11.7 (± 5.35) | 56 |
| *k*NN | EWT | 4 | No | 2100.3 (± 904.45) | 1661.9 (± 801.43) | 11.8 (± 5.36) | 30 |
| *k*NN | None | – | No | 2095.0 (± 912.16) | 1653.3 (± 804.72) | 11.7 (± 5.35) | 2 |
| *k*NN | STL-A | – | No | 2095.0 (± 912.16) | 1653.3 (± 804.72) | 11.7 (± 5.35) | 8 |
| SVR | CEEMDAN | 2 | No | 2791.2 (± 1282.6) | 2273.4 (± 1051.3) | 16.4 (± 4.38) | 3 |
| SVR | EEMD | 3 | No | 2807.0 (± 1303.8) | 2281.1 (± 1081.1) | 16.5 (± 4.39) | 3 |
| SVR | EMD | 2 | No | 2783.9 (± 1328.4) | 2267.3 (± 1097.1) | 16.2 (± 4.45) | 2 |
| SVR | EWT | 2 | No | 2948.3 (± 1200.7) | 2408.0 (± 991.79) | 17.8 (± 4.69) | 113 |
| SVR | None | – | No | 2833.1 (± 1221.5) | 2289.8 (± 1011.5) | 16.7 (± 4.04) | 4 |
| SVR | STL-A | – | No | 3041.3 (± 1004.7) | 2504.0 (± 835.30) | 19.1 (± 5.23) | 34 |
| XGBoost | CEEMDAN | 8 | No | 1605.7 (± 374.06) | 1240.8 (± 295.97) | 8.93 (± 2.19) | 3 |
| XGBoost | EEMD | 9 | No | 1512.3 (± 416.44) | 1156.0 (± 375.99) | 8.19 (± 2.54) | 51 |
| XGBoost | EMD | 9 | No | 1683.6 (± 367.17) | 1298.6 (± 323.66) | 9.29 (± 2.11) | 11 |
| XGBoost | EWT | 5 | No | 1740.0 (± 492.59) | 1368.6 (± 478.92) | 9.76 (± 3.17) | **1** |
| XGBoost | None | – | No | 1643.3 (± 470.69) | 1290.4 (± 449.28) | 9.26 (± 3.11) | 3 |
| XGBoost | STL-A | – | No | 1647.9 (± 507.71) | 1272.0 (± 454.99) | 9.16 (± 3.18) | **1** |
| | | | | | | **Total** | 425 |

for forecasting time series research. Hyperparameter tuning must be done carefully to avoid wasting hours of model training. Overfitting may explain why untuned models outperformed tuned models after 517 h of training model time. Thus, increasing the amount and variety of validation data and setting up cross-validation folds to cover more training data may solve the overfitting problem. For ONS dataset, decomposition methods outperformed non-decomposition methods. The test set results for ISO-NE dataset showed that the none decomposition method outperformed other models, but the error metrics were close. As mentioned, the tests were time-consuming due to many techniques and setups, but they yielded significant results. Using the optimal configuration, such as more training set coverage for hyperparameter tuning, and different combination ensembles of regression techniques for each decomposed IMF, the following tests may yield better results than this research.

Future research, as informed by the literature review on very short-term load forecasting (VSTLF), suggests that investigating deep recurrent models could be a valuable next step. The integration of these models into the proposed research framework is especially promising. Additionally, the literature indicates the potential benefits of leveraging stochastic multiobjective techniques for optimizing hyperparameters, which could enhance results. However, it is critical to note that both methods mentioned above might significantly elevate computational time. Lastly, to assess the versatility of this hybrid model, the proposed framework should be subjected to tests on various datasets.

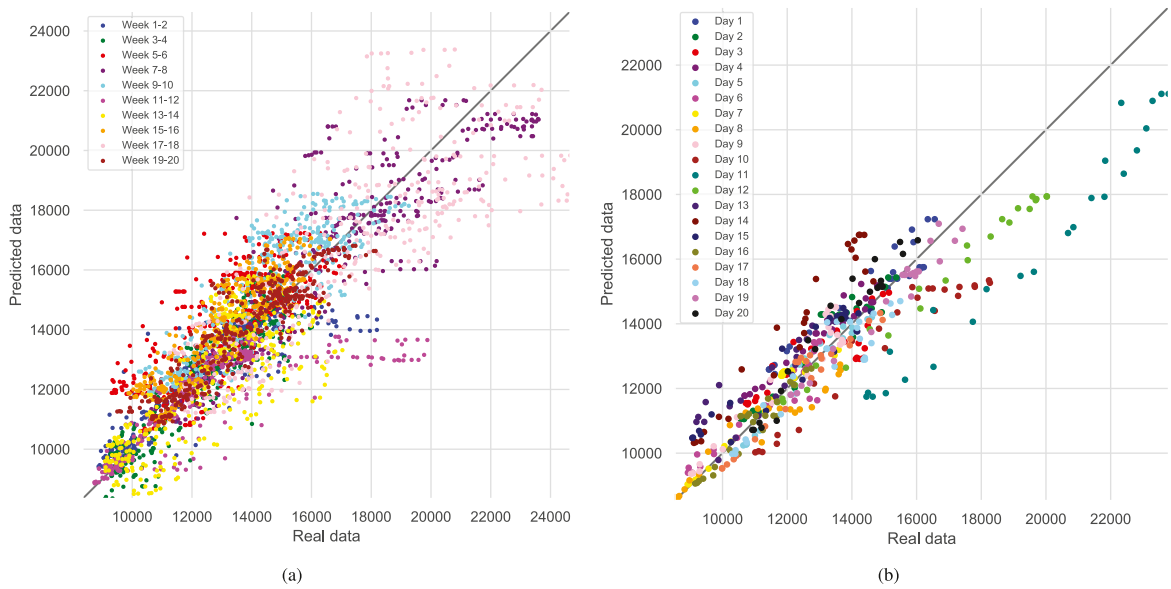



**Fig. 12.** ISO-NE dataset bias verification (a) of training set where each different color of scattered data represents the considered time (given a two weeks time window), (b) of the test set where each color of scattered data represents 1 of 20 days selected in 2019.

## CRediT authorship contribution statement

**Marcos Yamasaki Junior:** Software, Conceptualization, Methodology, Formal analysis, Validation, Writing – original draft, Writing – review & editing. **Roberto Zanetti Freire:** Supervision, Conceptualization, Writing – review & editing. **Laio Oriel Seman:** Writing – review & editing. **Stefano Frizzo Stefenon:** Writing – review & editing. **Viviana Cocco Mariani:** Writing – review & editing. **Leandro dos Santos Coelho:** Supervision, Conceptualization, Writing – review & editing.

## 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.

## Data availability

Data will be made available on request.

## Acknowledgments

The authors Mariani and Coelho would like to thank the National Council of Scientific and Technologic Development of Brazil - CNPq (Grants number: 307958/2019-1-PQ, 307966/2019-4-PQ, and 408164/2021-2-Universal), and Fundação Araucária PRONEX Grant 042/2018 for its financial support of this work. The author Freire would like to thank CNPq (Grant number: 312688/2021-0-PQ). The author Seman would like to thank the CNPq (Grants number: 404576/2021-4-Universal, 308361/2022-9-PQ). The author Yamasaki would like to thank the co-authors for their collaborative efforts and contributions to this paper. Acknowledgment is also due to the Pontifical Catholic University of Parana (PUCPR) for academic support and to Siemens for their support and flexibility, which made this work possible.

## References

[1] Sarkodie SA, Adams S. Electricity access, human development index, governance and income inequality in Sub-Saharan Africa. Energy Rep 2020;6:455–66.
[2] Hao P, Yin S, Wang D, Wang J. Exploring the influencing factors of urban residential electricity consumption in China. Energy Sustain Dev 2023;72:278–89.
[3] Wabukala BM, Bergland O, Rudaheranwa N, Watundu S, Adaramola MS, Ngoma M, Rwaheru AA. Unbundling barriers to electricity security in Uganda: A review. Energy Strategy Rev 2022;44:100984.
[4] Dao V, Ishii H, Takenobu Y, Yoshizawa S, Hayashi Y. Intensive quadratic programming approach for home energy management systems with power utility requirements. Int J Electr Power Energy Syst 2020;115:105473.
[5] Fan G-F, Zhang L-Z, Yu M, Hong W-C, Dong S-Q. Applications of random forest in multivariable response surface for short-term load forecasting. Int J Electr Power Energy Syst 2022;139:108073.
[6] Stefenon SF, Seman LO, Mariani VC, Coelho LdS. Aggregating prophet and seasonal trend decomposition for time series forecasting of Italian electricity spot prices. Energies 2023;16(3):1371.
[7] Sopelsa Neto NF, Stefenon SF, Meyer LH, Ovejero RG, Leithardt VRQ. Fault prediction based on leakage current in contaminated insulators using enhanced time series forecasting models. Sensors 2022;22(16):6121.
[8] Branco NW, Cavalca MSM, Stefenon SF, Leithardt VRQ. Wavelet LSTM for fault forecasting in electrical power grids. Sensors 2022;22(21):8323.
[9] Stefenon SF, Corso MP, Nied A, Perez FL, Yow K-C, Gonzalez GV, Leithardt VRQ. Classification of insulators using neural network based on computer vision. IET Gener, Transm Distrib 2021;16(6):1096–107.
[10] Ali M, Adnan M, Tariq M. Optimum control strategies for short term load forecasting in smart grids. Int J Electr Power Energy Syst 2019;113:792–806.
[11] Stefenon SF, Singh G, Yow K-C, Cimatti A. Semi-ProtoPNet deep neural network for the classification of defective power grid distribution structures. Sensors 2022;22(13):4859.
[12] da Silva RG, Moreno SR, Ribeiro MHDM, Larcher JHK, Mariani VC, dos Santos Coelho L. Multi-step short-term wind speed forecasting based on multi-stage decomposition coupled with stacking-ensemble learning approach. Int J Electr Power Energy Syst 2022;143:108504.
[13] Wang Y, Sun S, Chen X, Zeng X, Kong Y, Chen J, Guo Y, Wang T. Short-term load forecasting of industrial customers based on SVMD and XGBoost. Int J Electr Power Energy Syst 2021;129:106830.
[14] Zhang Q, Zhang J. Short-term load forecasting method based on EWT and IDBSCAN. J Electr Eng Technol 2020;15(2):635–44.
[15] Sankalpa C, Kittipiyakul S, Laitrakun S. Forecasting short-term electricity load using valiyeard ensemble learning. Energies 2022;15(22):8567.
[16] Duan Y. A novel interval energy-forecasting method for sustainable building management based on deep learning. Sustainability 2022;14(14):8584.
[17] Yu F, Wang L, Jiang Q, Yan Q, Qiao S. Self-attention-based short-term load forecasting considering demand-side management. Energies 2022;15(12):4198.
[18] Baliyan A, Gaurav K, Mishra SK. A review of short term load forecasting using artificial neural network models. Procedia Comput Sci 2015;48:121–5.
[19] Groß A, Lenders A, Schwenker F, Braun DA, Fischer D. Comparison of short-term electrical load forecasting methods for different building types. Energy Inf 2021;4(3):13.
[20] Yu Z, Yang J, Wu Y, Huang Y. Short-term power load forecasting under COVID-19 based on graph representation learning with heterogeneous features. Front Energy Res 2021;865.

[21] Sun J, Dong H, Gao Y, Fang Y, Kong Y. The short-term load forecasting using an artificial neural network approach with periodic and nonperiodic factors: A case study of Tai'an, Shandong Province, China. Comput Intell Neurosci 2021;2021:1–8.
[22] Genov E, Petridis S, Iliadis P, Nikopoulos N, Coosemans T, Messagie M, Camargo LR. Short-term load forecasting in a microgrid environment: Investigating the series-specific and cross-learning forecasting methods. J Phys Conf Ser 2021;2042(1):12035.
[23] Subbiah SS, Chinnappan J. An improved short term load forecasting with ranker based feature selection technique. J Intell Fuzzy Systems 2020;39(5):6783–800.
[24] Candela Esclapez A, López García M, Valero Verdú S, Senabre Blanes C. Automatic selection of temperature variables for short-term load forecasting. Sustainability 2022;14(20):13339.
[25] Stefenon SF, Seman LO, Aquino LS, dos Santos Coelho L. Wavelet-Seq2Seq-LSTM with attention for time series forecasting of level of dams in hydroelectric power plants. Energy 2023;274:127350.
[26] Klaar ACR, Stefenon SF, Seman LO, Mariani VC, Coelho LS. Structure optimization of ensemble learning methods and seasonal decomposition approaches to energy price forecasting in Latin America: A case study about Mexico. Energies 2023;16(7):3184.
[27] Bokde N, Feijóo A, Villanueva D, Kulat K. A review on hybrid empirical mode decomposition models for wind speed and wind power prediction. Energies 2019;12(2):254.
[28] Ali M, Prasad R. Significant wave height forecasting via an extreme learning machine model integrated with improved complete ensemble empirical mode decomposition. Renew Sustain Energy Rev 2019;104:281–95.
[29] Gao Y, Hang Y, Yang M. A cooling load prediction method using improved CEEMDAN and Markov Chains correction. J Build Eng 2021;42:103041.
[30] Klaar ACR, Stefenon SF, Seman LO, Mariani VC, Coelho LS. Optimized EWT-Seq2Seq-LSTM with attention mechanism to insulators fault prediction. Sensors 2023;23(6):3202.
[31] Wang Y, Wang D, Tang Y. Clustered hybrid wind power prediction model based on ARMA, PSO-SVM, and clustering methods. IEEE Access 2020;8:17071–9.
[32] Fan D, Sun H, Yao J, Zhang K, Yan X, Sun Z. Well production forecasting based on ARIMA-LSTM model considering manual operations. Energy 2021;220:119708.
[33] Liu K, Hu X, Zhou H, Tong L, Widanage WD, Marco J. Feature analyses and modeling of lithium-ion battery manufacturing based on random forest classification. IEEE/ASME Trans Mechatronics 2021;26(6):2944–55.
[34] Mussumeci E, Coelho FC. Large-scale multivariate forecasting models for Dengue-LSTM versus random forest regression. Spatial Spatio-Temporal Epidemiol 2020;35:100372.
[35] Chen Y, Zheng W, Li W, Huang Y. Large group activity security risk assessment and risk early warning based on random forest algorithm. Pattern Recognit Lett 2021;144:1–5.
[36] Bentéjac C, Csörgő A, Martínez-Muñoz G. A comparative analysis of gradient boosting algorithms. Artif Intell Rev 2021;54:1937–67.
[37] Zhang W, Wu C, Zhong H, Li Y, Wang L. Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization. Geosci Front 2021;12(1):469–77.
[38] Qiu Y, Zhou J, Khandelwal M, Yang H, Yang P, Li C. Performance evaluation of hybrid WOA-XGBoost, GWO-XGBoost and BO-XGBoost models to predict blast-induced ground vibration. Eng Comput 2021;1–18.
[39] Demir S, Sahin EK. An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost. Neural Comput Appl 2022;1–18.
[40] Corso MP, Perez FL, Stefenon SF, Yow K-C, Ovejero RG, Leithardt VRQ. Classification of contaminated insulators using k-nearest neighbors based on computer vision. Computers 2021;10(9):112.
[41] Stefenon SF, Bruns R, Sartori A, Meyer LH, Ovejero RG, Leithardt VRQ. Analysis of the ultrasonic signal in polymeric contaminated insulators through ensemble learning methods. IEEE Access 2022;10:33980–91.
[42] Zhou T, Thung K-H, Liu M, Shi F, Zhang C, Shen D. Multi-modal latent space inducing ensemble SVM classifier for early dementia diagnosis with neuroimaging data. Med Image Anal 2020;60:101630.
[43] Liu K, Li Y, Hu X, Lucu M, Widanage WD. Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries. IEEE Trans Ind Inf 2020;16(6):3767–77.
[44] Wong T-T, Yeh P-Y. Reliable accuracy estimates from k-fold cross validation. IEEE Trans Knowl Data Eng 2019;32(8):1586–94.
[45] Saud S, Jamil B, Upadhyay Y, Irshad K. Performance improvement of empirical models for estimation of global solar radiation in India: A k-fold cross-validation approach. Sustain Energy Technol Assess 2020;40:100768.
[46] Huang C-G, Huang H-Z, Li Y-F, Peng W. A novel deep convolutional neural network-bootstrap integrated method for RUL prediction of rolling bearing. J Manuf Syst 2021;61:757–72.
[47] Löning M, Bagnall AJ, Ganesh S, Kazakov V, Lines J, Király FJ. Sktime: A unified interface for machine learning with time series. CoRR 2019;abs/1909.07872. http://arxiv.org/abs/1909.07872.
[48] Stefenon SF, Kasburg C, Nied A, Klaar ACR, Ferreira FCS, Branco NW. Hybrid deep learning for power generation forecasting in active solar trackers. IET Gener, Transm Distrib 2020;14(23):5667–74.
[49] Divya D, Babu SS. Methods to detect different types of outliers. In: 2016 International conference on data mining and advanced computing (SAPIENCE). Ernakulam, India: IEEE; 2016, p. 23–8.
[50] Alghushairy O, Alsini R, Soule T, Ma X. A review of local outlier factor algorithms for outlier detection in big data streams. Big Data Cogn Comput 2020;5(1):1.
[51] Christen R, Mazzola L, Denzler A, Portmann E. Exogenous data for load forecasting: A review. In: Proceedings of the 12th international joint conference on computational intelligence. Budapest, Hungary: Science and Technology Publications; 2020, p. 489–500.
[52] Sun G, Chen T, Wei Z, Sun Y, Zang H, Chen S. A carbon price forecasting model based on variational mode decomposition and spiking neural networks. Energies 2016;9(1):54.
[53] Xie T, Zhang G, Hou J, Xie J, Lv M, Liu F. Hybrid forecasting model for non-stationary daily runoff series: A case study in the Han River Basin, China. J Hydrol 2019;577:123915.
[54] Voyant C, Notton G, Duchaud J-L, Almorox J, Yaseen ZM. Solar irradiation prediction intervals based on Box–Cox transformation and univariate representation of periodic autoregressive model. Renew Energy Focus 2020;33:43–53.
[55] Moon J, Park S, Rho S, Hwang E. A comparative analysis of artificial neural network architectures for building energy consumption forecasting. Int J Distrib Sens Netw 2019;15(9):1–19.