Automating Value-Oriented Forecast Model Selection by Meta-learning: Application on a Dispatchable Feeder

Dorina Werling(✉), Maximilian Beichter, Benedikt Heidrich, Kaleb Phipps, Ralf Mikut, and Veit Hagenmeyer

Karlsruhe Institute of Technology, Karlsruhe, Germany
dorina.werling@kit.edu

Abstract. To successfully increase the share of renewable energy sources in the power system and for counteract their fluctuating nature in view of system stability, forecasts are required that suit downstream applications, such as demand side management or management of energy storage systems. However, whilst many forecast models to create these forecasts exist, the selection of the forecast model best suited to the respective downstream application can be challenging. The selection is commonly based on quality measures (such as mean absolute error), but these quality measures do not consider the value of the forecast in the downstream application. Thus, we introduce a meta-learning framework for forecast model selection, which automatically selects the forecast model leading to the forecast with the highest value in the downstream application. More precisely, we use a meta-learning approach that considers the selection task as a classification problem. Furthermore, we empirically evaluate the proposed framework on the downstream application of a smart building's photovoltaic-battery management problem known as dispatchable feeder on building-level with a data set containing time series from 300 buildings. The results of our evaluation demonstrate that the proposed framework reduces the cost and improves the accuracy compared to existing forecast model selection heuristics. Furthermore, compared to a manual forecast model selection, it requires noticeably less computational effort and leads to comparable results.

Keywords: meta-learning · forecast value · forecast model selection

1 Introduction

The increasing share of decentralised, renewable energy sources challenges system operators since they must maintain system stability despite the uncertain

This project is funded by the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI, the Helmholtz Association under the Program "Energy System Design", and the German Research Foundation (DFG) as part of the Research Training Group 2153 "Energy Status Data: Informatics Methods for its Collection, Analysis and Exploitation".

and fluctuating behaviour of these energy sources. To maintain the system stability and to make optimal use of decentralised sources, smart grids make use of intelligent downstream applications. Such downstream applications include the intelligent management of smart buildings via demand side management [7,16], or dispatchable feeders [4,30,38]. These applications can be connected in a smart grid internet of things (IoT) environment, swiftly communicating with one another via information and communication technology (ICT) to ensure stable system operation. However, despite real-time data via smart meters, these downstream applications often rely on load and renewable energy generation forecasts.

In order to create these forecasts, a number of choices must be made regarding the forecast model, including the selection of the forecast method (e.g. autoregressive integrated moving average (ARIMA), support vector regression, or neural networks). Furthermore, several further decisions must be taken, such as the choice of the method's hyperparameters and the loss function used for training. Whilst many scientific papers promote certain forecast model choices that they claim will lead to good forecasts [2,18,23], it is important to consider and define what constitutes a "good" forecast. Thereby, the "goodness" of a forecast can be measured by its quality and/or its value [25]. The forecast quality evaluates the forecast solely from the forecaster's point of view and is typically measured by metrics such as mean absolute error (MAE) or mean squared error (MSE). On the other hand, the forecast value considers the downstream application's point of view by evaluating the performance of the downstream application based on the forecast. For the smart building optimisation examples described above, the performance of these downstream applications can be evaluated by calculating the economic costs based on the optimisation problem's result to assess the forecast value.

Several studies investigate the quality and the value of forecast methods [11,13,28] and hyperparameters [6,37] for solar, wind, and load forecasting. These studies show that the relation between forecast quality and value does not have to be linear or monotonic [25] and thus, improving the forecast quality may not always lead to a higher value in the downstream application in case of a forecast with remaining uncertainties. Additionally, [37] shows that the downstream application's setting and the considered data influence the forecast value. The authors show for a domestic photovoltaic-battery management problem that the battery capacity, as well as the prosumption profile, can also impact the forecast value.

These findings highlight the complexity of selecting forecast models to improve the value in the downstream application and indicate that the computational effort of a manual, optimal forecast model selection can be tremendous. To reduce the manual forecast model selection's computational burden while achieving comparable results, we propose a solution for automatically selecting forecast models based on the resulting forecast's value in the downstream application using meta-learning. Specifically, we propose a framework for forecast model selection that treats the selection task as a classification problem.

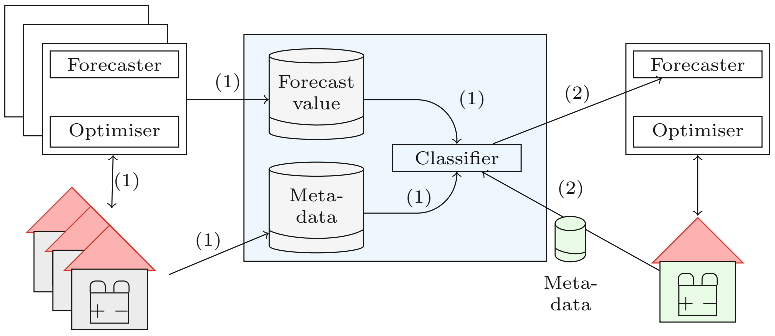


Fig. 1. The schematic representation of the proposed framework for a smart building optimisation as downstream application with the novelty marked in blue. In step (1), a classifier is trained using the buildings' metadata and the buildings' label of the forecast model leading to the forecast with the highest value. In step (2), the trained classifier can be operated to predict the forecast model leading to the highest value forecast for a new building utilising its metadata (marked in green). Then, the smart building optimisation can be executed. (Color figure online)

To achieve this, we train a classifier to select the forecast model leading to the forecast with the highest value in the downstream application, using metadata as input. In the smart building optimisation context, Fig. 1 displays the proposed framework. In step (1), the classifier is trained. For this training, we first execute the building optimisation with varying forecasts from different forecast models and then calculate the value of each forecast. Then, the forecast model leading to the forecast with the highest value is determined and together with metadata of the building used to train a classifier. In step (2), the trained classifier can then predict the forecast model leading to the forecast with the highest forecast value for a new building by utilising the new building's metadata. For evaluation, we apply our framework on the downstream application of a smart building's photovoltaic-battery-management problem - a dispatchable feeder - using real-world time series of 300 buildings.

The remainder of the paper is structured as follows. Section 2 gives a brief overview of the related work. Section 3 introduces the meta-learning framework for forecast model selection, while Sect. 4 presents the considered downstream application of a dispatchable feeder. Section 5 describes the experimental setup for evaluation and presents the corresponding results. Section 6 discusses the results and Sect. 7 wraps up the paper.

2 Related Work

In this section, we position our research compared to existing literature by addressing two aspects. First, we summarize prior work focusing on forecast models designed for the downstream application. Second, we present existing meta-learning approaches for quality-oriented forecast model selection.

One approach to designing forecast models based on the downstream application is to incorporate information from this downstream application into the forecast model. While [41] feeds back such information to the forecasting, [14,15] utilise the mathematical description of the downstream application, in this case an optimisation problem, during forecasting. However, these approaches can be computationally expensive and may not take into account all the relevant information influencing the forecast value. A simpler approach is to assume that the forecast value is solely dependent on the forecast error. Then, one can use the so-called cost-oriented loss function [19,22,24,36,40]. Thereby, the cost-oriented loss function is a piecewise function that assigns different weights to forecast errors, resulting in biased forecasts. While the form of the cost-oriented loss function needs to be known in [22,24,36,40] approximate the form in a computationally expensive manner. However, the cost-oriented loss function is not suitable for complex downstream applications with constraints, such as battery management problems. The approach of customising the loss function to fit the downstream application is also pursued in [1,20]. However, this approach is specifically designed for a single downstream application and cannot necessarily be generalised to create high-value forecasts for other applications.

Approaching the forecast model selection, several works aim to find the best-suited forecast method based on forecast quality measures using meta-learning [31,34,35]. Additionally, [9] directly predicts the RMSE of a forecast using meta-learning. However, these approaches do not consider the value of the forecast in the downstream application.

Summarizing, several existing works present approaches to either design forecasts based on their downstream applications or use meta-learning for forecast model selection with respect to forecast quality. In contrast, our meta-learning framework selects the forecast model with respect to the forecast value while not requiring knowledge of the downstream application during forecasting making it easily usable for various applications.

3 Meta-learning Framework for Forecast Model Selection

The idea of the underlying meta-learning framework for forecast model selection is to find the forecast model leading to the forecast with the highest value in the downstream application for a specific instance, e.g. a specific building. Thereby, the forecast value is a forecast evaluation metric that measures the performance of the downstream application based on the forecast and depends on the quantity of interest e.g. a building's electricity cost or self-sufficiency. Mathematically, the forecast value can be described by a function of the downstream application, its required data, and the information needed to generate the forecasts, namely the forecast model and the specific instance's data. The aim of the proposed framework is then to find the forecast model for each instance that maximises the forecast value, which we will refer to as the best forecast model in the following. Thus, we search for the best forecast model for instance $i \in \mathcal{I}$

$$f_i^\star = \underset{f \in \mathcal{F}}{\operatorname{argmax}} \, \text{Value}\Big(a, D_i, f\Big) \tag{1}$$

from a set of forecast models $\mathcal{F}$ given a downstream application a and the instance's data D_i. Thereby, D_i includes the input data of forecast models in $\mathcal{F}$ as well as additional data required by the downstream application a. Although this best forecast model may be found through a manual search, such a manual selection is cost-intensive and time-consuming. Therefore, we propose a meta-learning framework to identify the best forecast model automatically. In the following, we introduce this meta-learning framework for forecast model selection which interprets the selection task as a classification problem. First, we present the components of the proposed framework. Afterwards, we explain the usage of the framework including the training process and the operation.

3.1 Components of the Proposed Framework

The meta-learning framework for forecast model selection consists of four components. These are the set of forecasts models $\mathcal{F}$, the downstream application a, the metadata extraction component, and the classifier c.

The set of forecast models $\mathcal{F}$ comprises all considered forecast models. To handle the influence of the data D_i of different instances $i \in \mathcal{I}$ on what is the best forecast model, we require that the set of forecast models $\mathcal{F}$ is diverse. Thus, this diversity has to be ensured when $\mathcal{F}$ is created. The second component is the considered downstream application a. This downstream application requires a forecast provided by a forecast model $f \in \mathcal{F}$ as input for execution. The execution's performance determines the forecast value. The third component is the metadata extraction component. This component extracts the metadata m_i from D_i that is used by the classifier to determine the best forecast model. The last component is the classifier c. The task of the classifier is to select the forecast model leading to the forecast with the highest value in the downstream application a. Thus, we need to interpret the selection problem in Eq. (1) as a classification problem by considering each $f \in F$ as a class. Consequently, the target of the classification problem is the class of $f_i^\star$. Given the metadata m_i, the output of the classifier is then

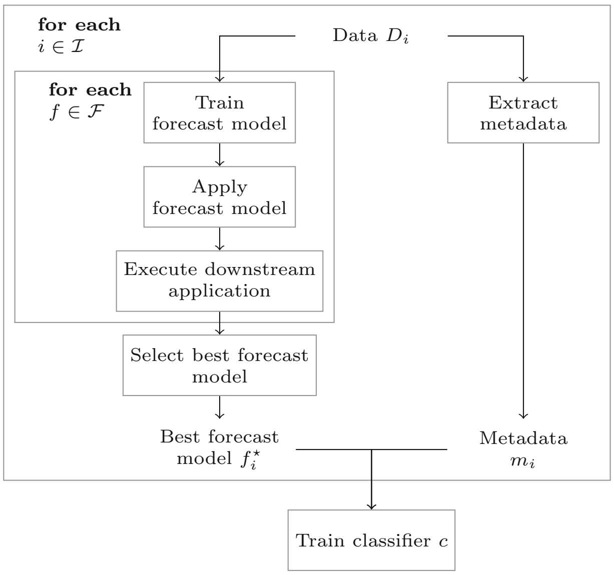
$$\hat{f}_i^\star = c(m_i).$$

3.2 Usage of the Proposed Framework

To use the proposed framework, we first need to train the framework in order to operate it afterwards.

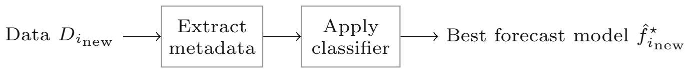
Step 1: Training Fig. 2a provides an overview of the proposed framework's training. This training requires the creation of the input features and the target variables. In the following, we present the creation of the input features and target variables as well as the training of the classifier in more detail.

First, to create the input features, we extract the metadata m_i of the data D_i for each instance $i \in \mathcal{I}$. Additionally, to create the target variables, we create a set of forecast models $\mathcal{F}$ and train the forecast models on the data D_i of each instance $i \in \mathcal{I}$. These forecast models provide forecasts to the downstream application a for execution. The execution's performance determines the forecast value for each $f \in F$ and $i \in \mathcal{I}$ from which we derive the target variable $f_i^\star$. Second, using these target variables and the corresponding metadata m_i, we train the classifier c.



for each $i \in \mathcal{I}$
Data D_i
for each $f \in \mathcal{F}$
Train forecast model
Extract metadata
Apply forecast model
Execute downstream application
Select best forecast model
Best forecast model $f_i^\star$
Metadata m_i
Train classifier c

(a) To train the proposed framework, training data needs to be created. Therefore, for the data D_i of each instance $i \in \mathcal{I}$ and each forecast model $f \in \mathcal{F}$ the forecast value in the downstream application is calculated. Based on this, the best forecast model $f_\mathrm{i}^\star$ for this instance is determined. Afterwards, $f_\mathrm{i}^\star$ is used as target variable together with the metadata m_i as input data to train the classifier c.



Data D_{i_new}
Extract metadata
Apply classifier
Best forecast model $\hat{f}_{i_\mathrm{new}}^\star$

(b) To operate the proposed framework, the metadata m_{i_new} of the data D_{i_new} of a new instance i_new is extracted. Based on this, the classifier determines the best forecast model $\hat{f}_{i_\mathrm{new}}^\star$. This forecast model can be trained and it's forecast provided to the downstream application a.

Fig. 2. The usage of the meta-learning framework for forecast model selection. (a) In the first step, the framework needs to be trained. (b) Afterwards, the framework can be operated.

Step 2: Operation. The operation of the proposed framework is displayed in Fig. 2b. First, for the new instance i_{new} and the corresponding data $D_{i_{\text{new}}}$, we extract the metadata $m_{i_{\text{new}}}$. Using this metadata as input, the classifier's output is the best forecast model $\hat{f}^{\star}_{i_{\text{new}}}$. Based on this output, the corresponding forecast model can be trained, and the resulting forecast provided to the downstream application a.

4 Applying the Proposed Framework: Application on a Dispatchable Feeder

In this section, we apply our framework to the downstream application of a dispatchable feeder as in [37]. We, therefore, first describe the dispatchable feeder before highlighting how we apply our framework to this downstream application.

4.1 Application Dispatchable Feeder

In this section, we introduce the exemplary downstream application on which we apply and evaluate our meta-learning framework for forecast model selection. The exemplary downstream application is a dispatchable feeder, which consists, from the system side, of an inflexible, volatile component and a flexible, but energy-constrained component [4,30]. The overall aim of the dispatchable feeder is to intelligently manage the flexible component so that the inflexibility inherent in the system is balanced out. Thereby, the management of the flexible component is described via a two-level non-linear optimisation problem. In the following, we first specify the considered system components, before we explain the management.

System Components of the Dispatchable Feeder. We apply a dispatchable feeder in a domestic building setting and consider the prosumption of the building with a rooftop photovoltaic (PV) panel as the inflexible and the domestic battery as the flexible component. To model these components, we consider a discrete system operation with time intervals indexed with $k \in \mathbb{N}$ and duration of $\Delta t \in \mathbb{R}$. Additionally, we consider only active power. To model the battery, we use its power output $P_s(k) \in \mathbb{R}$ and state of energy $E_s(k) \in \mathbb{R}$. Both variables are restricted by lower and upper bounds $\underline{P}_s, \overline{P}_s \in \mathbb{R}$ and $\underline{E}_s, \overline{E}_s \in \mathbb{R}_{\geq 0}$. Further, we model the dynamic evolution of the battery's state of energy via

$$E_s(k+1) = E_s(k) + \Delta t \cdot \left(P_s(k) - \mu P_s^+(k) + \mu P_s^-(k)\right) \tag{2}$$

with loss coefficient $0 \leq \mu \leq 1$ and $P_s^+(k) \in \mathbb{R}_{\geq 0}$ and $P_s^-(k) \in \mathbb{R}_{\leq 0}$ being the positive and negative parts of the battery's power output. The building's prosumption is defined as domestic load minus PV power generation. The power exchange between the grid and the dispatchable feeder is then the sum of the battery's power output and the building's prosumption.

Battery Management of the Dispatchable Feeder. In the following, we briefly introduce the battery management of the dispatchable feeder via a hierarchical optimisation problem with two levels. A detailed mathematical description of the optimisation problems can be found in Appendix A.1.

In the first level, we calculate a cost-minimal dispatch schedule $\tilde{P}_g(k) \in \mathbb{R}$ for the next day using deterministic forecasts of the prosumption $\hat{P}_l(k) \in \mathbb{R}$. As a cost function, we take peak shaving and self-consumption into account:

$$\begin{aligned} C_{DS}\big(\tilde{P}_g^+(k), \tilde{P}_g^-(k)\big) = c_\mathrm{q}^+ \cdot (\tilde{P}_g^+(k))^2 + c_\mathrm{l}^+ \cdot \tilde{P}_g^+(k) \\ + c_\mathrm{q}^- \cdot (\tilde{P}_g^-(k))^2 + c_\mathrm{l}^- \cdot \tilde{P}_g^-(k), \end{aligned} \tag{3}$$

with $\tilde{P}_g^+(k) \in \mathbb{R}_{\geq 0}$ and $\tilde{P}_g^-(k) \in \mathbb{R}_{\leq 0}$ being the positive and negative parts of the dispatch schedule and $c_\mathrm{q}^+, c_\mathrm{l}^+, c_\mathrm{q}^-, c_\mathrm{l}^- \in \mathbb{R}_{\geq 0}$ being weighting parameters.

In the second level, we minimize the deviation of the dispatch schedule $\Delta P_g(k) \in \mathbb{R}$ with consideration of the realised prosumption $P_l(k) \in \mathbb{R}$, while respecting the battery's technical constraints.

4.2 Applying the Proposed Framework

As described in Sect. 3, the proposed framework interprets the selection of the best forecast model as a classification task. In the downstream application of the dispatchable feeder, we are interested in identifying for each building the prosumption forecast model which provides the forecast with the highest value. As forecast value we consider the average daily total cost, with lower costs corresponding to a higher value [37]. Therefore, we briefly describe these costs below.

The total cost is comprised of two components: the cost associated with the dispatch schedule, as described in Eq. (3), and the cost resulting from the difference between the actual dispatch and the dispatch schedule, referred to as imbalance cost [4]. More precisely, we define the imbalance cost as

$$C_\mathrm{imb}\big(\Delta P_g(k)\big) = c_\mathrm{q}^\Delta \cdot \mid \Delta P_g(k) \cdot \Delta t \mid^2 + c_\mathrm{l}^\Delta \cdot \mid \Delta P_g(k) \cdot \Delta t \mid$$

with $\Delta P_g(k)$ being the difference between the actual dispatch and the dispatch schedule and the weighting parameters c_q^Δ and c_l^Δ[1].
The total cost can then be expressed as

$$C_{total}\big(\tilde{P}_g^+(k), \tilde{P}_g^-(k), \Delta P_g(k)\big) = C_{DS}\big(\tilde{P}_g^+(k), \tilde{P}_g^-(k)\big) + \alpha \cdot C_{imb}\big(\Delta P_g(k)\big)$$

with imbalance cost factor[2] α. Further, we sum the total cost over 24 h and average this daily total cost over the considered days to obtain the average daily total cost[3].

5 Evaluation

We evaluate the meta-learning framework for forecast model selection on the previously described downstream application of a dispatchable feeder.

[1] In this paper, we select $c_\mathrm{q}^\Delta = 0.05 \frac{€}{\mathrm{kWh}^2}$ and $c_\mathrm{l}^\Delta = 0.3 \frac{€}{\mathrm{kWh}}$ as in [5].
[2] In this paper, we select as imbalance cost factor either 2 or 10 as in [5].
[3] In the following, we use the average daily total cost without its unit (€).

5.1 Experimental Setup

This section presents the experimental setup[4] for the evaluation of the proposed framework.

Forecast Models. Driven by the results of [37], we consider neural networks using varying loss functions as forecast models. More precisely, we use three-layered fully connected neural networks. As input, the neural networks receive the historical prosumption of the past 24 h. The hidden layer has 16 neurons and a ReLU [17] activation function. The output layer provides the prediction of the prosumption for the next 42 h using 42 output neurons and a linear activation function. Furthermore, we use 20% of the training data for validation to apply early stopping, a batch size of 512, and the RMSProb optimiser.

To achieve a diverse set of forecast models, we consider four different loss functions. This set of forecast models is guaranteed to be diverse because the used loss functions are responsible for the resulting forecast's properties. Thereby, a loss function quantifies how well the neural network models the true values $y_t, t \in [1 \dots N]$, and is minimised in training. The first loss function is the **mean absolute error** (MAE). It is the mean over the absolute errors and defined as

$$\mathrm{MAE} = \frac{1}{N}\sum_{t=1}^{N} |y_t - \hat{y}_t| \,.$$

The MAE treats large and small errors equally. Second, the **mean squared error** (MSE) is the mean over the sum of the squared errors and defined as

$$\mathrm{MSE} = \frac{1}{N}\sum_{t=1}^{N} (y_t - \hat{y}_t)^2.$$

Due to the squared error term, the loss function is more sensitive to outliers. Third, the **Huber** loss function combines properties of the MSE and the MAE and is defined as

$$\mathrm{Huber} = \frac{1}{N}\sum_{t=1}^{N} \begin{cases} \frac{1}{2}(y_t - \hat{y}_t)^2, & \text{for } |y_t - \hat{y}_t| \leq 1 \\ (|y_t - \hat{y}_t| - \frac{1}{2}), & \text{otherwise} \end{cases}.$$

Therefore, if the absolute of the error is less than one, the squared error loss is used, and if not, an absolute error based loss is used. This error function is therefore less influenced by outliers than the MSE. The **pinball** loss, which is often utilised for generating $\tau \in (0, 1)$ quantile forecasts, has the propensity to produce biased estimates. Specifically, the true values are underestimated with a probability of τ and overestimated with a probability of $\tau - 1$. It is defined as

$$\mathrm{pinball}(\tau) = \frac{1}{\mathrm{N}}\sum_{\mathrm{t}=1}^{\mathrm{N}} \max(\tau \cdot (\mathrm{y_t} - \hat{\mathrm{y}}_\mathrm{t}), (\tau - 1) \cdot (\mathrm{y_t} - \hat{\mathrm{y}}_\mathrm{t})).$$

In the present paper, we choose τ values of 0.1, 0.25, 0.75, and 0.9.

[4] See Appendix A.2 for a detailed description of the implementation.

Classifier Setup. The classifier within our proposed framework requires metadata that captures relevant time series information as input. Thus, we consider a set of statistical features as input features including the mean, standard deviation, minimum, 25th percentile, median, 75th percentile, maximum, skewness, and kurtosis of the prosumption time series. Furthermore, we also include the average daily prosumption profile, i.e. the mean over all considered days for each hour. For further analysis with different input features, the reader is referred to Appendix A.4. To reduce the dimensionality of the input data, we use SKLearn's principal component analysis with 70% of explained variance [26]. Therefore, we scale the input data with SKLearn's standard scaler.

Given the selected metadata, we evaluate our framework with six different classifiers to cover a broad range of classification approaches including tree-, distance-, support vector-, and neural network-based classifiers, which we briefly describe in the following. The first classifier is the XGBoost classifier [10]. XGBoost boosts multiple decision trees iteratively to improve the prediction. The second classifier is the k-nearest neighbour (kNN) [12]. It determines the k nearest neighbours for a given test sample. The final classification is then performed by a majority vote of the k nearest neighbours. The third classifier is the support vector classifier (SVC) [27]. The SVC aims to find a hyperplane between two classes [32]. To apply the SVC in the multi-class scenario of the considered downstream application, the one-vs-one strategy is used. The fourth classifier is the multi-layer perceptron (MLP). The MLP consists of one or more hidden layers of fully connected neurons with a non-linear activation function to approximate arbitrary functions. The fifth classifier is a decision tree (DT) [8]. DTs extract rules from the training data. Based on these rules, DTs predict the class of the given sample. The sixth classifier is the naive Bayes (NB) [39]. NB assumes the conditional independence of the input features and uses them for the prediction of a conditional probability given the prior probability of the output variable.

Data. For our evaluation, we use the "Ausgrid - Solar home electricity data" set [29]. The data set contains load and PV power generation time series from 300 residential buildings in Australia spanning three years from 1st July 2010 to 30th June 2013 in a 30 min resolution. We resample the data to hourly resolution and calculate the prosumption data by subtracting the PV power generation from the load. To compare different ratios of load with regards to the installed PV power generation, we scale the PV power generation with the factors 1, 5, and 10, and the load with factors 1/5, 1/2, 1, 2, and 5.

Further, we use training and test splitting, see Appendix A.3. Thereby, for training the neural networks, we use the first two years of data as training data set. For evaluating the neural networks, we utilise the last year as test data set. Further, for training the classifiers, we use the first 200 buildings. For evaluating the classifiers, we employ the last 100 buildings. Additionally, for all buildings, we extract the metadata based on the first two years and consider the output labels based on the last year.

Benchmarks. To evaluate the performance of our proposed framework, we compare it to two benchmarks. The first benchmark is the **one loss function** benchmark. As suggested by the name, this benchmark applies one forecast model to all buildings. More precisely, we train one neural network with the same loss function for each building on the first two years. As the second benchmark, we consider a **manually selected loss function** for each building. In this benchmark, we calculate which loss function is cost-minimal for each building based on the first two years of data. Then, each building applies this cost-minimal loss function for the last year.

Metrics. For evaluation, we use three metrics. The first metric is the F_1 score as an accuracy measure for the considered classifier. The F_1 score is defined as

$$\mathrm{F}_1 = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}},$$

with TP being the true positives, FP being the false positives and FN being the false negatives. For the one loss function benchmark, the F_1 score corresponds to the percentage of buildings for which the loss function is cost-minimal. Second, to measure the forecast value in our downstream application we use the average daily total cost in Sect. 4.2, which take the imbalance and the dispatch schedule cost into account. Thereby, we calculate the mean of the building's average daily total cost over the considered buildings for imbalance cost factors 2 and 10. Third, we measure computational effort by recording the average computation time of each component in seconds and calculate based on this the average forecast model selection time in seconds. Thereby, the latter consists of the time required to select the forecast model and to generate the forecast.

5.2 Results

In this section, we evaluate the performance of the meta-learning framework for forecast model selection. First, we compare the cost and accuracy of our framework with the two selected benchmarks. Second, we evaluate the impact of different classifiers on the forecast model selection performance and, finally, address the computational effort.

Benchmarking. We compare the cost and accuracy of our framework to the two benchmarks with respect to imbalance cost factors 2 and 10 in Table 1.

Starting with imbalance cost factor 2, we observe for the one loss function benchmark that the selected loss function has a noticeable impact on the cost and accuracy. This benchmark achieves the lowest cost (6.09) with MAE as loss function. However, similar costs are obtained when using Huber (6.11) and MSE (6.27). The corresponding accuracies with MSE, MAE, and Huber is 0.25, 0.22, and 0.17 respectively. In contrast, pinball 0.75 results in cost of 8.83 despite being the cost-minimal loss function for the most buildings (namely 34%). In comparison, the proposed framework with SVC as classifier reduces the one loss

Table 1. The average daily total costs and the F_1 scores of the proposed framework and the considered benchmarks for the imbalance cost factors 2 and 10. The metrics are calculated for the test data set with the last year of data and the last 100 buildings. Note, for the average daily total costs lower values are better and for the F_1 scores higher values.

Approaches	Imbalance cost factor 2		Imbalance cost factor 10	
	Average daily total costs (€)	F_1 scores	Average daily total costs (€)	F_1 scores
One loss function with				
MAE	6.09	0.22	20.05	0.16
MSE	6.27	0.25	18.95	0.20
Huber	6.11	0.17	19.05	0.11
Pinball 0.10	9.62	0.00	43.78	0.00
Pinball 0.25	7.41	0.02	30.47	0.01
Pinball 0.75	8.83	0.34	27.04	0.37
Pinball 0.90	18.13	0.00	60.08	0.16
Manually selected loss function	5.92	0.69	17.67	0.73
Proposed framework (SVC)	5.93	0.68	17.80	0.67

function benchmark cost by at least 2.6% to 5.93 and improves the accuracy to 0.68. However, the proposed framework has slightly higher cost (5.92) and lower accuracy (0.69) compared to the manually selected loss function benchmark.

For imbalance cost factor 10, the one loss function benchmark reaches its lowest cost (18.95) using MSE as loss function, with similar costs using Huber (19.05) and MAE (20.05). The corresponding accuracy is 0.20 with MSE, 0.16 with MAE, and 0.11 with Huber. The highest accuracy of 0.37 is reached with pinball 0.75. In comparison, the proposed framework with SVC as classifier, again, reduces the one loss function benchmark cost by at least 6% to 17.80 and improves the accuracy to 0.67. Further, similar to imbalance cost factor 2, the difference in cost and accuracy between the proposed framework and the manually selected loss function benchmark is less than 1% and 10% respectively.

Impact of Classifier. To investigate the impact of the classifier, we compare the performance of the proposed framework with six classifiers. Based on the results for the imbalance cost factors 2 and 10 in Table 2, we present three observations.

First, for with imbalance cost factor 2, we observe that the proposed framework achieves the lowest cost of 5.93 when using SVC and MLP. The classifiers kNN, XGBoost, and decision tree lead to costs of 6.00, 6.03 and 6.07 respectively. With respect to the accuracy, the order of the classifier is similar with SVC/MLP (0.68), XGBoost (0.64), kNN (0.63), and decision tree (0.58). In contrast, naive Bayes leads to the highest cost of 6.21 as well as to the lowest accuracy of 0.6.

Second, for imbalance cost factor 10, SVC and MLP lead to the lowest costs of 17.80 and 17.81 and to accuracy of 0.67. Further, the classifier kNN, XGBoost, and naive Bayes lead to costs of 17.98, 18.13, and 18.21 and accuracies of 0.63,

Table 2. The average daily total costs and the F_1 scores of the proposed framework using different classifiers for the imbalance cost factors 2 and 10. The metrics are calculated for the test data set with the last year of data and the last 100 buildings. Thereby, we calculate the mean over five runs with the values in the brackets being the minimum and maximum. Note, for the average daily total costs lower values are better and for the F_1 scores higher values.

Approaches	Imbalance cost factor 2		Imbalance cost factor 10	
	Average daily total costs (€)	F_1 scores	Average daily total costs (€)	F_1 scores
XGBoost	6.03	0.64	18.13	0.61
kNN	6.00	0.63	17.98	0.63
SVC	5.93	0.68 (0.67,0.68)	17.80	0.67
MLP	5.93 (5.92,5.95)	0.68 (0.67, 0.68)	17.81 (17.76, 18.39)	0.67
Decision tree	6.07 (6.06,6.09)	0.58 (0.57,0.59)	18.35 (18.32, 18.39)	0.56 (0.55,0.57)
Naive Bayes	6.21	0.6	18.21	0.58

0.61, and 0.58 respectively. In contrast to the results for the imbalance cost factor 2, decision tree performs worst with cost of 18.35 and accuracy of 0.56.

Our final observation regarding the classifiers' impact is that each classifier reaches a higher accuracy for imbalance cost factor 2 compared to imbalance cost factor 10.

Computational Effort. We first measure each component's computation time. Afterwards, we calculate the time of the proposed framework and the benchmarks to select the forecast model for a new building.

For each component, Table 3a provides the average computation time in seconds per building. The most time-intensive component is the optimisation problem's run time on the first two years, followed by the neural network's training time. The other components require a negligible amount of time.

Based on the measured components' times, we can estimate the forecast model selection time for a new building. For the one loss function benchmark, the neural network with the considered loss function must be trained on the first two years of the new building's data. For the manually selected loss function benchmark, we need to train neural networks for each loss function, generate forecasts with the trained neural networks and solve the optimisation problem with the resulting forecasts for the first two years. For the proposed framework, we must first extract the metadata based on the first two years, then run the classifier, and, finally, train the neural network with the selected loss function once. Table 3a shows the resulting forecast model selection times. In this table, we make two observations. First, the manually selected loss function benchmark has, noticeably, the highest forecast model selection time with 287.14 s. The one loss function benchmark and the proposed framework require noticeably less time. Second, we observe that despite using a meta-learning approach the forecast model selection time of the proposed framework is only slightly higher than that of the one loss function benchmark.

Table 3. The average computation time of each component and the average forecast model selection time of the proposed framework and the considered benchmarks in seconds for a new building. Note, the times do not depend on the considered imbalance cost factor.

(a) Computation time of the components

Components	Computation time (s)
Classifier inference time	
XGBoost	0.02
kNN	0.08
SVC	0.10
MLP	0.01
Decision tree	0.00
Naive Bayes	0.01
Metadata generation time	0.19
Forecasting NN training time	9.38
Forecasting NN inference time	0.07
Optimisation problem run time	31.57

(b) Forecast model selection time of the proposed framework and the considered benchmarks for a new building.

Approaches	Forecast model selection time (s)
One loss function	9.38
Manually selected loss function	287.14
Proposed framework (SVC)	9.67

6 Discussion

This section discusses the previously reported results in Sect. 5.2, the benefits, and the limitations of the meta-learning framework for forecast model selection.

With regard to the results of the evaluation, we discuss three aspects. First, the results indicate that the proposed framework reduces the cost and improves the accuracy compared to the one loss function benchmark. Thereby, the choice between the classifiers SVC and MLP does not affect the proposed framework's performance. In contrast, for this application, using the naive Bayes and decision tree classifier is not recommended. Furthermore, the performance of the proposed framework with respect to cost and accuracy is comparable to the performance of the manually selected loss function benchmark. However, there is still potential for further improvement, e.g. more advanced classifier as well as hyperparameter optimisation. Second, we observe a non-monotonic, non-linear relation between cost and accuracy. More precisely, improving the accuracy does not necessarily lead to lower cost. For the one loss function benchmark, this means that the cost-minimal loss function for most buildings is not necessarily the cost-minimal loss function for the whole data set. This observation can be explained by extensive costs for the remaining buildings and highlights the complexity of selecting the loss function with respect to the forecast value in the downstream application. Finally, the results show that the proposed framework reduces the computational effort compared to the manually selected loss function benchmark by 97%. That makes the proposed framework particularly interesting for downstream applications for which scalability is essential. For example, in the considered application of the dispatchable feeder, scalability and low computational effort could become important for the joint optimisation of multiple buildings. Additionally, in contrast to the manually selected loss function benchmark, the computational effort of the proposed framework increases negligibly when the set of forecasting models is expanded. Besides the benefit with respect to scalability, we want to highlight a further advantage that comes from the design of the proposed framework. This is, due to its simplicity, it can be easily applied to various applications without extensive knowledge of the optimisation problem.

With regard to the proposed framework's limitations, we discuss three aspects. First, it should be noted that the proposed framework has been evaluated on one application. Second, the proposed framework currently uses two years of historical data for the initial training process. Therefore, further evaluation is required to determine how a reduced training set affects the performance. Third, the proposed framework selects the forecast model for each building once. Further research could extend the framework to an online setting that continually re-classifies each building based on recent information. This extension is especially motivated by the results of the manually selected loss function benchmark that indicates that the forecast value of a forecast model might change over time.

7 Conclusion

The present paper addresses the complexity of selecting a forecast model based on the resulting forecast's value in the downstream application. To automate this selection and avoid a computational expensive manual selection, we propose a meta-learning framework for forecast model selection. More precisely, we consider the selection task as a classification problem and train a classifier based on labels referring to the forecast model which provides the forecast with highest value in the downstream application. We evaluate this meta-learning framework for forecast model selection on the exemplary downstream application of a smart building's domestic photovoltaic-battery management problem known as dispatchable feeder. Therefore, we consider neural networks with different loss functions as forecast models. Thus, the selection task is to select the neural network's loss function providing forecasts with the highest value for the dispatchable feeder. The results show that our framework reduces the cost and improves the accuracy compared to selecting the same loss function for each building. In comparison with selecting the loss function for each building manually, the proposed framework leads to similar cost and accuracy requiring noticeably less computational effort.

In future work, we plan to expand our framework by incorporating the dynamically selection of the forecast model based on recent input data and by increasing the considered set of forecast models.

A Appendix

A.1 Optimisation Problems

In the following, the first and second level optimisation problems of the dispatchable feeder in Sect. 4.1 are described. In the first level, the cost function in Eq. (3) is minimised under consideration of the system constraints. The first level optimisation problem can be described by

$$\begin{aligned}
&\min_{\{X\}_\mathcal{K}} \sum_{k\in\mathcal{K}} C_{DS}\big(\tilde{P}_g^+(k), \tilde{P}_g^-(k)\big) \\
&\text{s.t. for all } k \in \mathcal{K} \\
&\qquad (2) \\
&E_s(k_0) = E_s^0 \\
&\tilde{P}_g(k) = P_s(k) + \hat{P}_l(k) \\
&\tilde{P}_g(k) = \tilde{P}_g^+(k) + \tilde{P}_g^-(k) \\
&\tilde{P}_g^-(k) \leq 0 \\
&P_s(k) = P_s^+(k) + P_s^-(k) \\
&P_s^+(k) \geq 0 \\
&P_s^-(k) \leq 0 \\
&0 = P_s^+(k) \cdot P_s^-(k) \\
&\underline{P}_s \leq P_s(k) \leq \overline{P}_s \\
&\underline{E}_s \leq E_s(k) \leq \overline{E}_s
\end{aligned} \tag{4}$$

with a discrete scheduling horizon $\mathcal{K}$, decision vector $X(k) = \big(\tilde{P}_g(k),\ \tilde{P}_g^+(k), \tilde{P}_g^-(k), E_s(k+1), P_s(k),\ P_s^+(k), P_s^-(k)\big)^T$, and parameters $\hat{P}_l(k)$, E_s^0, $\underline{P}_s, \overline{P}_s, \underline{E}_s, \overline{E}_s$. Thereby, the state of energy at the start of scheduling $k_0 \in \mathbb{N}$ has to be known or estimated.

In the second level, the deviation of the dispatch schedule $\Delta P_g(k) \in \mathbb{R}$ is minimised while considering the realised prosumption $P_l(k) \in \mathbb{R}$ and the battery's technical constraints. It can be described by

$$\begin{aligned}
&\min_{X(k)} \quad \big(\Delta P_g(k)\big)^2 \\
&\qquad (2) \\
&E_s(k) = E_s^k \\
&P_g(k) = P_s(k) + P_l(k) \\
&P_g(k) = \tilde{P}_g(k) + \Delta P_g(k) \\
&P_s(k) = P_s^+(k) + P_s^-(k) \\
&P_s^+(k) \geq 0 \\
&P_s^-(k) \leq 0
\end{aligned} \tag{5}$$

$$
\begin{aligned}
0 &= P_s^+(k) \cdot P_s^-(k) \\
\underline{P}_s \leq & P_s(k) \leq \overline{P}_s \\
\underline{E}_s \leq & E_s(k) \leq \overline{E}_s
\end{aligned}
$$

with actual dispatch $P_g(k) \in \mathbb{R}$, decision vector $X(k) = \big(\Delta P_g(k), P_g(k), E_s(k+1), P_s(k), P_s^+(k), P_s^-(k)\big)^T$ and parameters $\tilde{P}_g(k)$, $P_l(k)$, E_s^k, $\underline{P}_s, \overline{P}_s$, $\underline{E}_s, \overline{E}_s$. Thereby, the state of energy in $k \in \mathbb{N}$ is known.

A.2 Implementation

In the following, we briefly describe the hard- and software and the optimisation problems' parameter specification used for our evaluation.[5]

All of our experiments are performed on a small server with 32 cores (2.1GHz), 64GB RAM, and a Nvidia Titan RTX. To ensure reproducibility and reusability, we implement the experiments using the pyWATTS library [21]. To solve the optimisation problem in our downstream application, we use the Python version of CasADi [3] with IPOPT [33]. We set the parameters of the optimisation problem in Eq. (4) and Eq. (5) as in [5], see Table 4. Further, we implement all classifiers except XGBoost with implementation provided by SKLearn [26]. For XGBoost, we use the XGBoost library [10]. Furthermore, we use the default hyperparameters for all classifiers apart from the MLP classifier, where we raise the maximum number of epochs to 1000.

Table 4. Parameter specification of the optimisation problems in Eq. (4) and Eq. (5).

Parameter	Value
Δt	1 (hour)
$\mathcal{K}$	$\{k_s, \ldots, k_s + 29\}$[a]
c_q^+	0.05 (€/kWh^2)
c_l^+	0.3 (€/kWh)
c_q^-	0.05 (€/kWh^2)
c_l^-	0.15 (€/kWh)
$\underline{P}_s$	-5 (kW)
$\overline{P}_s$	5 (kW)
$\underline{E}_s$	0 (kWh)
$\overline{E}_s$	13.5 (kWh)
μ	0.05
E_s^0	day 1: 6 (kWh)
	days 2 - 7: estimated[b]

[a] $k_s \in \mathbb{N}$ is the index of the time interval starting at midnight
[b] The initial state of energy E_s^0 for days two to seven is estimated via an optimisation problem, see [4].

[5] The Code is publicly available at https://github.com/KIT-IAI/Automating-Value-Oriented-Forecast-Model-Selection-by-Meta-Learning.

A.3 Training and Test Data Sets

Fig. 3 displays the training and test data set splitting for the forecast models and the classifiers.

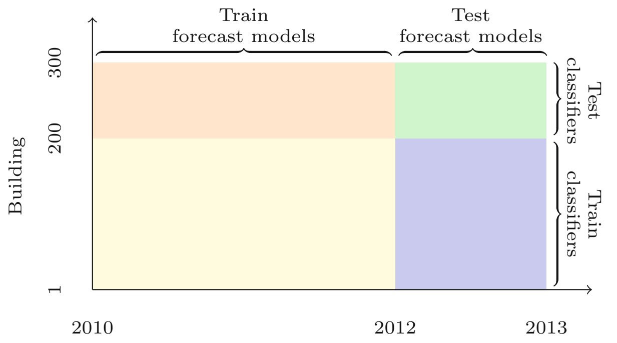


Fig. 3. The training and test data sets for the forecast models and the classifiers. Note, for the classifiers, we extract the metadata from the first two years and consider the output labels based on the last year.

A.4 Input Features

Table 5 shows the costs and accuracies of the proposed framework for different input features.

Table 5. The average daily total costs (left) and the F_1 scores (right) of the proposed meta-learning framework for forecast model selection using different classifiers and different input features for imbalance cost factor 2. Thereby, the metadata is calculated for the prosumption time series and consists of the following statistical features: [B] mean, standard deviation, minimum, 25th percentile, median, 75th percentile, and maximum; [F1] mean over all days for each hour; [F2] mean, minimum, and maximum of each day; [F3] mean over all days of minimum and maximum for each day; [F4] seasonality and trend; [F5] skewness and kurtosis; [F6] autocorrelation. The metrics are calculated for the test data set with the last year of data and the last 100 buildings. Thereby, we calculate the mean over five runs with the values in the brackets being the minimum and maximum. Note, for the average daily total costs lower values are better and for the F_1 scores higher values.

Input features	Classifier											
	XGBoost		kNN		SVC		MLP		Decision Tree		Naive Bayes	
	(€)	F_1 scores	(€)	F_1 scores	(€)	F_1 scores	(€)	F_1 scores	(€)	F_1 scores	(€)	F_1 scores
B+F1	6.02	0.6	5.99	0.62	5.93	0.68	5.93 (5.93, 5.94)	0.68	6.04 (6.03, 6.05)	0.55 (0.55, 0.56)	6.22	0.6
B+F1+F2	6.0	0.6	5.98	0.62	5.93	0.68	5.96 (5.94, 5.99)	0.66 (0.66, 0.67)	6.1 (6.08, 6.11)	0.56 (0.56, 0.57)	6.27	0.57
B+F1+F3	6.04	0.61	5.99	0.63	5.93	0.68	5.93 (5.92, 5.94)	0.68 (0.68, 0.69)	6.09 (6.07, 6.1)	0.55 (0.54, 0.56)	6.15	0.6
B+F1+F4	6.02	0.61	5.99	0.61	5.93	0.68	5.93 (5.92, 5.94)	0.68 (0.68, 0.69)	6.07 (6.06, 6.07)	0.55 (0.55, 0.56)	6.22	0.6
B+F1+F5	6.03	0.64	6.00	0.63	5.93	0.68	5.93 (5.92, 5.95)	0.68 (0.67, 0.68)	6.07 (6.06,6.09)	0.58 (0.57, 0.59)	6.21	0.6
B+F1+F6	6.03	0.63	5.98	0.63	5.93	0.68	5.93 (5.92, 5.93)	0.69 (0.68, 0.69)	6.05 (6.04, 6.07)	0.56 (0.55, 0.57)	6.21	0.6
B+F1+F2+F5	5.99	0.62	5.98	0.61	5.93	0.68	5.95 (5.94, 5.98)	0.67 (0.66, 0.68)	6.09 (6.08, 6.11)	0.56 (0.55, 0.57)	6.27	0.57
B+F1+F3+F5	6.01	0.64	5.98	0.64	5.93	0.68	5.94 (5.92, 5.97)	0.68 (0.67, 0.7)	6.06 (6.05, 6.06)	0.58 (0.58, 0.59)	6.21	0.61
B+F1+F2+F3+F5	5.99	0.62	5.98	0.61	5.93	0.68	5.96 (5.94, 5.97)	0.67 (0.66, 0.67)	6.09 (6.09, 6.11)	0.56 (0.55, 0.58)	6.27	0.57
B+F1+F2+F3+F4+F5+F6	5.99	0.61	5.98	0.61	5.93	0.68	5.96 (5.95, 5.97)	0.66 (0.66, 0.67)	6.1	0.56	6.27	0.57

References

1. Abdulla, K., Steer, K., Wirth, A., Halgamuge, S.: Improving the on-line control of energy storage via forecast error metric customization. J. Energy Storage **8**, 51–59 (2016)
2. Ahmad, T., Zhang, H., Yan, B.: A review on renewable energy and electricity requirement forecasting models for smart grid and buildings. Sustain. Urban Areas **55**, 102052–102082 (2020)
3. Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M.: CasADi - a software framework for nonlinear optimization and optimal control. Math. Program. Comput. **11**, 1–36 (2019)
4. Appino, R.R., González Ordiano, J.Á., Mikut, R., Faulwasser, T., Hagenmeyer, V.: On the use of probabilistic forecasts in scheduling of renewable energy sources coupled to storages. Appl. Energy **210**, 1207–1218 (2018)
5. Appino, R.R., González Ordiano, J.Á., Mikut, R., Hagenmeyer, V., Faulwasser, T.: Storage scheduling with stochastic uncertainties: feasibility and cost of imbalances. In: 2018 Power Systems Computation Conference (PSCC), pp. 1–7 (2018)
6. Bessa, R.J., Miranda, V., Botterud, A., Wang, J.: 'Good' or 'bad' wind power forecasts: a relative concept. Wind Energy **14**(5), 625–636 (2011)
7. Biyik, E., Kahraman, A.: A predictive control strategy for optimal management of peak load, thermal comfort, energy storage and renewables in multi-zone buildings. J. Build. Eng. **25**, 100826–100836 (2019)
8. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth International Group, Belmont, CA (1984)
9. Carneiro, D., Guimarães, M., Carvalho, M., Novais, P.: Using meta-learning to predict performance metrics in machine learning problems. Expert Syst. **40**(1), e12900 (2023)
10. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. Association for Computing Machinery, New York, NY, USA (2016)
11. Coignard, J., Janvier, M., Debusschere, V., Moreau, G., Chollet, S., Caire, R.: Evaluating forecasting methods in the context of local energy communities. Int. J. Electr. Power Energy Syst. **131**, 106956 (2021)
12. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
13. David, M., Boland, J., Cirocco, L., Lauret, P., Voyant, C.: Value of deterministic day-ahead forecasts of PV generation in PV + storage operation for the Australian electricity market. Sol. Energy **224**, 672–684 (2021)
14. Donti, P.L., Amos, B., Kolter, Z.: Task-based end-to-end model learning in stochastic optimization. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp 5490–5500. Curran Associates Inc., Red Hook, NY, USA (2017)
15. Elmachtoub, A.N., Grigas, P.: Smart "predict, then optimize." Manage. Sci. **68**(1), 9–26 (2022)
16. Frahm, M., et al.: Occupant-oriented economic model predictive control for demand response in buildings, pp. 354–360. e-Energy '22, Association for Computing Machinery, New York, NY, USA (2022)
17. Fukushima, K.: Cognitron: a self-organizing multilayered neural network. Biol. Cybern. **20**(3–4), 121–136 (1975). https://doi.org/10.1007/BF00342633

18. Ghofrani, M., Alolayan, M.: Time series and renewable energy forecasting, pp. 77–92 (2017)
19. Granger, C.W.J.: Prediction with a generalized cost of error function. OR **20**(2), 199–207 (1969)
20. Haben, S., Ward, J., Vukadinovic Greetham, D., Singleton, C., Grindrod, P.: A new error measure for forecasts of household-level, high resolution electrical energy consumption. Int. J. Forecast. **30**(2), 246–256 (2014)
21. Heidrich, B., et al.: pyWATTS: python workflow automation tool for time series (2021). arXiv:2106.10157
22. Khabibrakhmanov, I., Lu, S., Hamann, H.F., Warren, K.: On the usefulness of solar energy forecasting in the presence of asymmetric costs of errors. IBM J. Res. Dev. **60**(1), 7:1–7:6 (2016)
23. Kuster, C., Rezgui, Y., Mourshed, M.: Electrical load forecasting models: a critical systematic review. Sustain. Urban Areas **35**, 257–270 (2017)
24. Li, G., Chiang, H.D.: Toward cost-oriented forecasting of wind power generation. IEEE Trans. Smart Grid **9**(4), 2508–2517 (2018)
25. Murphy, A.H.: What is a good forecast? an essay on the nature of goodness in weather forecasting. Weather Forecast. **8**(2), 281–293 (1993)
26. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
27. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv. Large Margin Classif. **10**(3), 61–74 (1999)
28. Putz, D., Gumhalter, M., Auer, H.: The true value of a forecast: assessing the impact of accuracy on local energy communities. Sustain. Energy Grids Networks **33**, 100983 (2023)
29. Ratnam, E.L., Weller, S.R., Kellett, C.M., Murray, A.T.: Residential load and rooftop PV generation: an Australian distribution network dataset. Int. J. Sustain. Energ. **36**(8), 787–806 (2017)
30. Sossan, F., Namor, E., Cherkaoui, R., Paolone, M.: Achieving the dispatchability of distribution feeders through prosumers data driven forecasting and model predictive control of electrochemical storage. IEEE Trans. Sustain. Energy **7**(4), 1762–1777 (2016)
31. Talagala, T.S., Hyndman, R.J., Athanasopoulos, G.: Meta-learning how to forecast time series. Monash Econometrics and Business Statistics Working Papers 6/18, Monash University, Department of Econometrics and Business Statistics (2018)
32. Vapnik, V.N.: The Nature of Statistical Learning Theory, 2nd edn. Springer, New York (2000). https://doi.org/10.1007/978-1-4757-3264-1
33. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006)
34. Wang, C., Bäck, T., Hoos, H.H., Baratchi, M., Limmer, S., Olhofer, M.: Automated machine learning for short-term electric load forecasting. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 314–321. IEEE, Xiamen, China (2019)
35. Wang, X., Smith-Miles, K., Hyndman, R.: Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. Neurocomputing **72**(10), 2581–2594 (2009)
36. Wang, Y., Wu, L.: Improving economic values of day-ahead load forecasts to real-time power system operations. IET Gener. Trans. Distrib. **11**(17), 4238–4247 (2017)

37. Werling, D., Beichter, M., Heidrich, B., Phipps, K., Mikut, R., Hagenmeyer, V.: The impact of forecast characteristics on the forecast value for the dispatchable feeder. In: Companion Proceedings of the 14th ACM International Conference on Future Energy Systems, pp. 59–71. e-Energy '23 Companion, Association for Computing Machinery, New York, NY, USA (2023)
38. Werling, D., Heidrich, B., Çakmak, H.K., Hagenmeyer, V.: Towards line-restricted dispatchable feeders using probabilistic forecasts for PV-dominated low-voltage distribution grids. In: Proceedings of the Thirteenth ACM International Conference on Future Energy Systems, pp. 395–400. e-Energy '22, Association for Computing Machinery, New York, NY, USA (2022)
39. Zhang, H.: The optimality of naive Bayes. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004). AAAI Press, Miami Beach, Florida, USA (2004)
40. Zhang, J., Wang, Y., Hug, G.: Cost-oriented load forecasting. Electric Power Syst. Res. **205**, 107723 (2022)
41. Zhao, C., Wan, C., Song, Y.: Cost-oriented prediction intervals: on bridging the gap between forecasting and decision. IEEE Trans. Power Syst. **37**(4), 3048–3062 (2022)