SimBuild
2024

Eleventh National Conference of IBPSA-USA
Denver, Colorado
May 21 – 23, 2024

Autonomous Load Forecast Framework with Dynamic Model Selection

Christoph Gehbauer[1,3,*], Nicholas DeForest[1], Peter Grant[1], Manfred Tragner[2]
José Baptista[3], and Douglas R. Black[1]
[1]Lawrence Berkeley National Laboratory, Berkeley, CA
[2]University of Applied Sciences JOANNEUM, Graz, Austria
[3]University of Trás-os-Montes and Alto Douro, Vila Real, Portugal
[*]Corresponding author. E-mail: cgehbauer@lbl.gov

Abstract

The accurate forecasting of weather conditions and electricity demand is of great importance in smart building and distributed energy resource operations. It ensures efficient resource allocation, operational cost reduction, and mitigation of environmental impacts. Traditional forecast methods often fall short due to their reliance on simplistic statistical techniques and limited adaptability to complex, dynamic patterns. Machine learning has emerged as a powerful but complex approach for improving the accuracy of such predictions by leveraging advanced algorithms. In this context, a framework which hosts a publicly available library of traditional and state-of-the-art machine learning models tailored for weather and electricity demand forecasting in buildings is proposed. Models are dynamically selected and combined based on an internal optimization algorithm to autonomously operate without user interaction and to ensure optimal forecast performance over the lifetime of the installation.

Introduction

In an era defined by environmental consciousness and technological advancement, the smart control of buildings has emerged as a promising tool to reduce energy costs and combat climate change through appropriate management of electricity demand. These innovative smart control systems harness cutting-edge technology to seamlessly integrate various aspects of building management, from lighting and cooling to comfort and electricity consumption. By doing so, they not only enhance the occupant comfort and health (Dong et al. 2019), (Merabet et al. 2021), and (Clark et al. 2019), but are also able to shift electricity demand (Gehbauer, Lee, and Wang 2023), (Lee et al. 2015), and (Mariano-Hernández et al. 2021) to align with periods of high renewable generation and therefore low marginal greenhouse gas (GHG) emissions from electricity production (Deetjen and Azevedo 2019). Lowering electricity demand during peak periods and shifting consumption to off-peak hours also relieves stress on the electric power grid (Tang, Wang, and Li 2021) and reduces overall energy expenses (Szinai et al. 2020). Smart building control systems, therefore, represent a crucial step towards a sustainable future where architecture and technology harmoniously contribute to our global commitment to a greener, more resilient planet.

The suite of available smart building control concepts and devices is vast, ranging from simple rule based controls (Park et al. 2019) for single devices, over smart devices and appliances, to cutting edge predictive control systems (Drgoňa et al. 2020) and (Gehbauer et al. 2020). However, this orchestration of the interdependent relationships between building loads and electricity generation to reduce overall GHG emissions is non trivial. For example, one flavor of predictive control systems is model predictive control (MPC) where a mathematical formulation is employed to model the building's thermal and electrical characteristics, occupants, heating ventilation and air conditioning (HVAC), and other systems. Weather forecast data over a defined horizon of typically 24 hours is used to predict the building's electricity demand and, in conjunction with a numerical solver, optimized to meet various objectives, e.g., energy efficiency, energy cost, greenhouse gas emissions. This proactive approach not only promotes energy conservation but also helps align when and how much electricity is used and therefore better aligns demand with low carbon electricity supply. However, smart control systems require knowledge of when internal and external disturbances such as environmental conditions, e.g., outdoor air temperature, solar irradiation, or wind, and occupancy patterns. Forecasts are needed to predict these disturbances on a granular level and to provide the appropriate inputs to the smart building control system. But forecasts are not only used for control systems. Forecasting of occupancy patterns and determining when and which spaces are utilized can provide a strategic advantage for building managers to optimize energy usage, fine-tune climate controls, and ensure occupant comfort, all

while reducing operational costs (Esrafilian-Najafabadi and Haghighat 2022).

Typical forecast models are powered by advanced analytics and machine learning algorithms (Zhang et al. 2021). Depending on the specific applications, many of these models are proprietary and embedded in control systems. Others are open-source software at varying state of development and testing. While proprietary forecast models often offer seamless integration with a manufacturer's control system, they also require a large upfront investment in data collection and model development, making them cost-prohibitive for smaller organizations. Additionally, the maintenance of these models can be complex and expensive. They may also lack the adaptability and flexibility of open-source solutions when it comes to integrating with other building management systems or evolving alongside other grid-interactive applications. On the other hand, open-source developments provide greater flexibility to be utilized for various applications, but often require specialized knowledge to setup and maintain.

One such set of forecast tools are regression-based machine learning algorithms (Sarker 2021). They are used across various industries and domains to gain insights and predict numerical outcomes based on input variables. They involve the mathematical modeling of the relationships between a set of independent and dependent variables, allowing for the estimation and forecasting of continuous values. Various regression models exist, each with unique strengths and suitability for specific types of data and problem domains. Linear regression, for example, is a fundamental model, assuming a linear relationship between the input variables and the output (Montgomery, Peck, and Vining 2021). Polynomial regression extends this linearity by accommodating polynomial relationships. And decision trees offer a versatile and intuitive model, able to capture complex relationships through a series of hierarchical decisions (Breiman 2001).

Selecting the appropriate forecast models that are best suited for a given dataset, such as occupancy patterns or electricity consumption, is a critical step in smart building control. Each forecast model includes a set of hyperparameters which describe the learning algorithm used to train the underlying forecaster and strongly influence the performance of the model. Since control systems heavily rely on forecasts, the choice of the wrong model or hyperparameters can greatly reduce or eliminate their effectiveness (Schratz et al. 2019). To achieve accurate and reliable results, it's important to consider the complexity of the underlying dynamic system. Building occupancy patterns and electricity consumption are often influenced by a multitude of factors, including external weather conditions, seasonal variations, and unforeseen events. Moreover, the volume and quality of data can greatly vary between sites, making it challenging to find robust patterns and trends.

This work seeks to greatly simplify the utilization of weather, load, and occupancy forecasting in buildings by (i) utilizing existing open-source forecast models and algorithms to assemble a library of forecast models specifically tailored to real-time application in building control, (ii) introducing a framework to dynamically select the best model based on the currently available data, (iii) increasing resiliency for cybersecurity and unforeseen events by selecting fallback models which do not require external inputs, and (iv) requiring minimal user interaction for autonomous operation over the lifetime of the installation.

Methodology

The methodology for developing the proposed forecast framework with dynamic model selection consists of several key steps and components, which are described in more detail in the following subsections:

- Library Architecture: The foundation of the framework involves designing and developing a modular library structure that can seamlessly integrate various forecasting models, accommodate diverse data sources, and ensure flexibility and scalability in adding new forecasting techniques.
- Model Selection and Integration: The library incorporates a selection of traditional (e.g., regression-based, auto-regressive) and machine learning models (e.g., random forest, neural network, gradient boosting machines). Each model is integrated to function seamlessly within the library.
- Preprocessing: Preprocessing tools are integrated to handle data cleaning, missing value imputation, and data transformation, ensuring consistent input data.
- Model Training and Selection: The forecast library automates the process of model training and hyperparameter selection. An internal optimization logic allows for autonomous and dynamic selection of the best model for the given data.
- Experimental Evaluations: The library's efficacy is demonstrated through experimental evaluation using real-world datasets from buildings and structures at a U.S. military facility.

Example Application

Camp Parks is a 2,500 acre military installation located in Dublin, California, and serves as a multi-functional base to support training, logistics, and intelligence operations. Its strategic location in the San Francisco Bay Area makes it a critical asset for military readiness and emergency response in the region. The base is home to various facilities, more than 120 buildings consuming 8 GWh of electricity annually, including administrative buildings, barracks, training areas, maintenance shops, and other essential structures. The size and layout of specific buildings and facilities varies, and includes modern infrastructure designed to support the diverse needs of the units and personnel stationed there.

Forecast Models

The forecast model library was assembled with best of class and diverse algorithms which resulted in a set of currently eleven different forecast models which are based on six fundamental algorithms. Each fundamental algorithm leverages a different learning technique: random forest (scikit-learn), extra trees (scikit-learn), gradient boosting (scikit-learn), multi layer perceptron (scikit-learn), auto-regressive (statsmodels), and time-of-week-temperature (in-house). The Appendix includes more information on each of the models and training techniques. The source of each model is shown in parentheses, and includes scikit-learn[1] (Pedregosa et al. 2011), statsmodels[2] (Seabold and Perktold 2010), and in-house developed tools (Vrettos and Gehbauer 2019).

Forecast Model Hyperparameter Tuning

Each forecast model has a set of parameters, typically referred to as hyperparameters, that are defined before the training process starts and control various aspects of the learning algorithm's behavior. Examples for hyperparameters are the learning rate, number of training epochs, batch size, architecture of neural networks, or activation functions. Properly tuning hyperparameters is crucial for optimizing a model's performance, but requires an involved process that includes understanding the impacts of the different hyperparameters, choosing several values to explore for each hyperparameter, training and testing the forecaster to evaluate the performance, and finally selecting the hyperparameter set which provides the best result.

To support this tuning process and optimize performance on any in-use dataset, the forecast library contains an automated *tune_hyperparameters* function. This function leverages scikit-learn's *GridSearchCV* and *RandomizedSearchCV* functions to identify sets of hyperparameters with best Root Mean Square Errors (RMSE) and enable repeated re-tuning when deployed in controllers. The function returns a list containing an instance of each forecast model with the highest performing hyperparameters. Repeated tuning of hyperparameters over time also allows customizing the forecasters to yield optimal performance as the training dataset grows over time.

Initial Forecast Model Setup

Some of the forecast tools from the library are customized to improve performance. Customization focuses on either 1) creation of pipelines including data preprocessing steps, 2) tuning the hyperparameters of the forecaster for more accurate or faster predictions, or 3) creating pipelines that enable single target forecasters to be used for multi target problems. The pipelines and/or hyperparameters used in the library were identified using either the TPOT[3] (Le, Fu, and Moore 2020) or a parametric study evaluating the performance of different hyperparameter combinations on data sets from prior projects (Gehbauer, Black, and Grant 2023). In each case the goal of the hyperparameter search was to return low RMSE values when comparing the predictions to the testing dataset.

The established pipelines from TPOT are recreated in the forecast library, and variations of the base models using these pipelines are indicated by the suffix *pipeline*. For example, the library contains a regressor called *extra_trees_pipeline* which is an implementation of scikit-learn's *extra_trees* leveraging the *PolynomialFeatures* data preprocessing function. The parametric study used a dataset of the electric load from the Camp Parks military base to tune the hyperparameters of multiple forecasters. Forecasters using hyperparameters identified in this process are indicated with the *tuned* prefix. To provide some forecaster tools with faster computation time, the hyperparameter sets yielding an RMSE within 10% of the best set in the least time are recreated and indicated using the term *fast*. For example, the *tuned_fast_mlp* forecaster is an implementation of scikit-learn's multi-layer perceptron forecaster with hyperparameters tuned to closely match the data from the Camp Parks military base.

[1] An open-source Python package for machine learning: https://scikit-learn.org/

[2] An open-source Python package for statistical computations: https://www.statsmodels.org/

[3] The Python base Tree-based Pipeline Optimization Tool (TPOT): https://github.com/EpistasisLab/tpot

Orchestrating Framework

The dynamics of building energy forecasting can be difficult to intuit and are often influenced by the specific characteristics of a building: mechanic systems, building envelope construction, weather and climate, occupancy schedule, etc. Consequently, forecasting techniques that perform well in one application may be entirely unsuitable for another. Furthermore, understanding the reasons for these disparities in forecaster performance typically requires detailed examination of datasets that may be difficult or costly to collect. The model selection framework introduced in this study intends to assist users in avoiding this particular challenge.

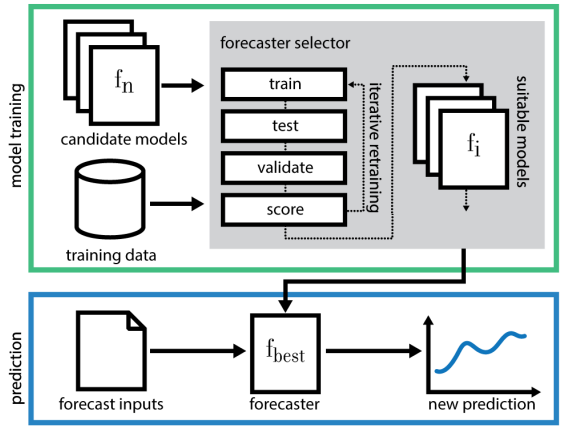

Figure 1: Architecture of the autonomous forecast framework with its dynamic model selection. During model training (green outline) a set of candidate forecast models (f_n) is supplied to the forecast selector framework (shaded in gray). Models (f_i) are evaluated based on the training data and the most suitable model is selected. The resulting best forecast model (f_{best}) can then be used for regular prediction (blue outline) given user inputs. The training is periodically repeated (e.g., every day or every week) indicated by the iterative training arrow.

The dynamic forecast framework, shown in Figure 1, is designed to use the "bucket of models" technique of ensemble learning. Within this method, a set of defined model candidates are trained and validated using the same underlying historical dataset. Based on their relative performance to a user-defined scoring metric, RMSE in this study, the model with the best score is selected and used to generate a prediction. This approach has the benefit of applying the best suited model for a given problem, without the user necessarily understanding the specific rational driving that outcome. Since this approach uses the best-scoring model directly in generating a prediction, its overall performance is only limited to the best-performing model in the candidate set.

The framework automates the following steps in order to generate a prediction:

- Data preprocessing: training data is input into the framework via a number of input streams and automatically formatted into feature (X) and target (y) datasets.
- Candidate model instantiation: candidate models from the library are instantiated and prepared for training and validation. Model instantiation is implemented so that model training can be performed as parallel processes to reduce total training time.
- Training of candidate models: training data is divided into training and testing portions. Candidate models are trained and assessed under a user-defined scoring metric (e.g., RMSE). Cross-validation of candidate models can also be selected by user, wherein multiple train and test splits are performed and average performance scores are generated.
- Model validation: To ensure that trained models exhibit adequate performance, users can specify minimum thresholds for the specified scoring metric (e.g., maximum allowable RMSE) before it can be selected by the framework. Minimum training dataset size can also be specified to ensure valid results.
- Model selection: Based on the training method and scoring metric defined by the user the suitable models are ranked and a single best model is selected by the framework. This trained model instance is then employed to generate predictions until the next iteration of the framework training and selection is executed.

The framework employs additional features to enhance its value in an ongoing forecasting application. In addition to reading historical training data from an existing source (e.g. time-series data file or database), the framework can dynamically manage incoming data streams, then process and add these new observations to its training dataset. Because training and tuning of candidate forecast models can be slow and computationally expensive, the functions of data management, model training, and new prediction generation can be separated into separate parallel compute instances. Parallelization of these

two functions allows the overall tool to continuously update training dataset, increase the frequency with which the candidate models are retrained and selected, and generate new forecast outputs with the latest data and best performing models. The two instances can be defined as:

- Data management loop: Receives data from external sources, manages training dataset, writes dataset to external file/database at user-specified triggers, reads trained forecast models from external file, and generates the forecast with the selected best model.
- Model training loop: Reads training data generated by data loop, trains candidate models when triggered, selects best performing model, and writes model objects to external file.

Several of the candidate forecast models described above require input data (e.g. weather forecasts) that must be collected from an external source. Should this data collection fail, such candidate models may be unable to operate until communication to external sources is restored. To account for this risk, candidate models include an attribute to indicate whether they require forecast data as an input. In the event of failure to collect these required data, the forecaster selector framework can filter the list of candidate models to only those that can operate without external data, which can be for example auto-regressive such as sarimax.

Dynamic Model Selection

An example of using the dynamic forecast framework code in the programming language Python is provided in Listing 1. Users can import a list of default forecast models from the forecaster library. This list includes the forecaster class objects as well as the default parameters used to instantiate each model. Both the list and parameter settings can be modified by users as needed. Users must then provide a training dataset, consisting of a data-frame of independent, feature variables, and a series of corresponding target variables. Next, a dictionary of additional input parameters must be provided to the framework. Here, default values can be imported and used directly, or modified by the user. Each of these objects is then passed into an instance of the *ForecasterFramework* class. Once instantiated, users can directly call the *evaulateAll* method to train and assess the set of the candidate models. If the resulting trained models satisfy the minimum specified performance threshold, then the best model can be used to generate new predictions by passing a data-frame containing new feature variables into the predict method of the framework class instance.

Listing 1: Python example.

```
# import forecaster library
from fcDyn import (
    ForecastFramework,
    defaultParams,
    exampleData
)

# define user preferences (optional)
params = defaultParams.copy()
params['min_days'] = 6

# instantiate forecast framework
fc = ForecasterFramework(
    params=params,
    data=exampleData
)

# evaluate candidate forecasters
# select best model for predictions
fc.evaluateAll()

# predict with best model
fc.predict(exampleData['X'])
```

Results

The functionality of the introduced forecast framework is demonstrated by first analyzing the performance of individual forecast algorithms at different points in time and consequent data availability, and second by showcasing an example application with a real building controller to reduce a site's electricity bill over a period of three months.

The performance of the pool of forecasters predicting the building load at Camp Parks military facility for three different scenarios (8, 17, and 82 days after installation) is shown in Figure 2. Each forecaster is re-trained after a selected duration of the installation, which is defined by the scenario, and then a prediction is generated for the following 24 hours. Since forecasts are only evaluated at specific points of time, no intermittent re-training was conducted in this analysis. The real data represents the shape of a typical office load with reduced demand during the night and ramped-up demand during the day. It differs from a typical office demand given Camp Park's 24/7 operation scheme and associated occupation and electricity demand during the night, where electricity demand is reduced from the 150 kW daytime peak to 100

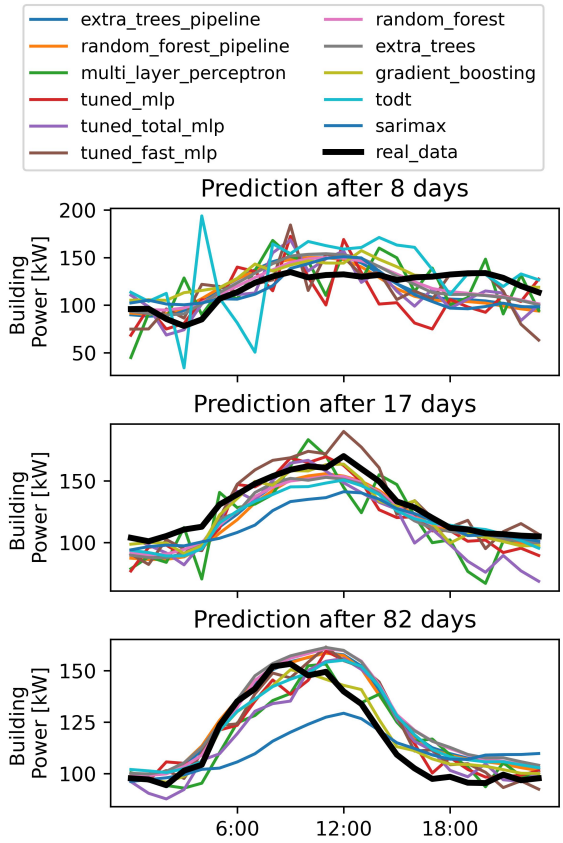

Figure 2: Examples of forecast generation for a building load (y-axis), in kW, at Camp Parks highlighting the distribution of forecast models (colored lines) and real data (bold black line) for one day (x-axis). The first subplot shows the results after 8 days of operation, the second subplot shows the results after 17 days, and the third subplots shows the results after 82 days of operation.

kW nighttime demand. It can be seen that the forecasters are able to capture the trend of higher demand during the daytime and lower, steady demand during the night. However, just after installation (first subplot - 8 days of operation) the spread of forecasters and random outliers is high, given the small sample size for training. Progressing to the next subplot after about two weeks (17 days) of operation all forecasters are able to improve their prediction, and outliers become closer spread. Last, after about three months (82 days) of operation all but the *sarimax* model are able to provide reasonable forecasts.

The resulting statistics and metrics for the pool of forecast models at the three month (82 days) timeline are shown in Table 1. The table is separated in true (i.e., evaluated during post processing to analyze the forecaster's real world performance given the full dataset) and predicted (i.e., without knowledge of future data trends) metrics for the coefficient of determination (r2) and RMSE. The predicted RMSEs range from 11.1 to 18.5 kW. While most forecasters are within the 12 kW range (e.g., *extra_trees*, *gradient_boosting*, *random_forest*, and *todt*) the *sarimax* model performs the worst with the highest RMSE. This is also reflected in the true RMSE column, where *sarimax* also shows the highest RMSE of 145.5 kW. The true RMSEs also reveal the difficulty to predict performance without knowledge of future data trends. For example, while the *gradient_boosting* model's performance is estimated at similar level for predicted and true RMSE, 11.2 and 13.5 kW respectively, most other models greatly over-estimate forecast accuracy, manifested by significantly lower RMSEs during prediction versus true. In this context, the *random_forest_pipeline* shows best performance with a predicted RMSE of 11.1 kW during operation, but yields an RMSE of 47.9 kW when evaluated with the full dataset. However, overall trends of forecaster performance are preserved and predicted sufficiently accurate to dynamically select either the best or one of the best performing forecast models form the library. For the introduced example of *random_forest_pipeline*, while the true RMSE was significantly higher, it still yielded second best performance for the full dataset, providing accurate and reliable forecasts during in-situation operation.

Table 1: Example error metrics for one day after 82 days of operation. Each row shows one of the eleven forecast models from the library.

	r2 true	rmse pred	rmse true
extra_trees	0.7	11.5	72.3
extra_trees_pipeline	0.8	11.3	53.3
gradient_boosting	0.9	11.2	13.5
multi_layer_perceptron	0.8	15.7	51.0
random_forest	0.7	11.2	69.5
random_forest_pipeline	0.8	11.1	47.9
sarimax	0.4	18.5	145.5
todt	0.8	12.8	53.9
tuned_fast_mlp	0.8	15.3	49.9
tuned_mlp	0.8	14.5	49.3
tuned_total_mlp	0.7	14.0	66.8

In order to illustrate the behavior of forecast models over time, Figure 3 shows the progression of five core forecast model errors over the time span of three months,

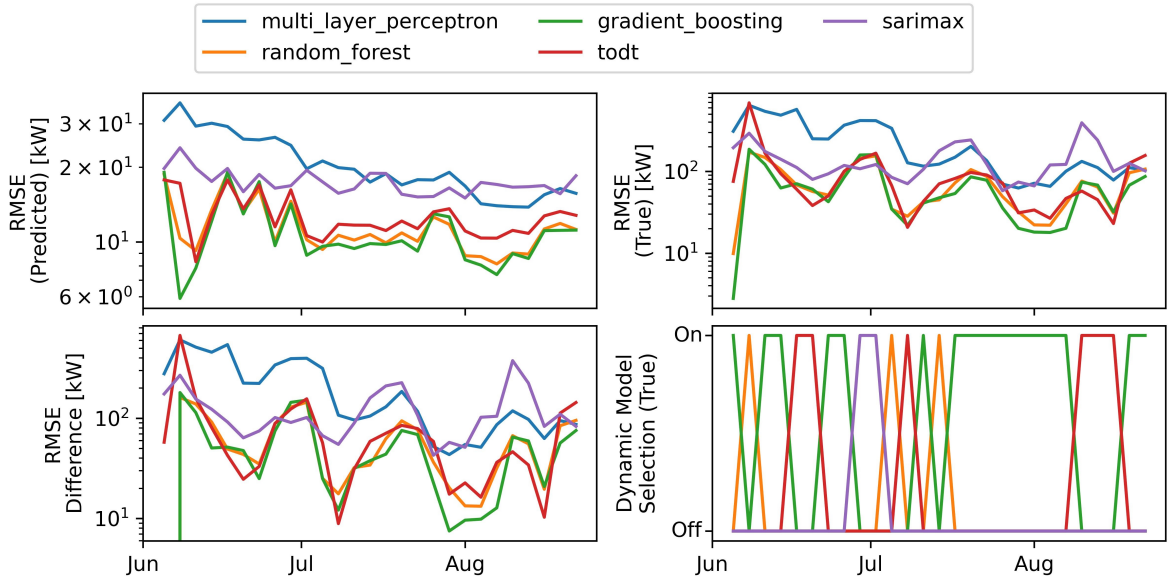

Figure 3: Example application for three summer months (x-axis; June through August) showing RMSE progression and model selection as new data is observed, and forecasters are retrained. Results are shown for one forecast model per base algorithm (colored lines). The first subplot on the upper left shows the progression of RMSE, in kW, for predictions where future data is not known. The second subplot on the upper right shows the progression of RMSE for true predictions over the following five days. The third subplot on the lower left shows the difference between the predicted and true RMSEs. The last subplot on the lower right indicates the currently used forecast model for the true predictions of the dynamic model selector, setting each forecast model to On or Off based on the determined minimal RMSE.

from June through August. It can be seen that forecast errors decrease over time, as more data is observed and used for training. Similar to the previous example illustrated in Table 1 the differences between predicted and true RMSE are significant (shown in the third subplot on the lower left) but both follow a similar trend and equivalent distribution of error across the five forecast models. The last subplot on the lower right illustrates the dynamic model selection process, where each forecast model is encoded with an on/off flag, and only one forecast model is selected at any given time. It can be seen that at the beginning of the deployment forecast models are often alternated, reflecting the underlying dynamics of the loads to be predicted and the size of observed and diverse data samples. Each of the forecast models is selected at one point in time, which results in greater and more reliable forecasts.

To further analyze the expected errors with the dynamic model selection, Table 2 summarizes and analyzes the RMSE results for each training period, which is set to three days in this study. For both, the predicted and true errors, the difference between a conventional time-of-day-temperature model and the dynamically selected combined model are significant, 16% and 40% respectively. Improvements are also achieved when selecting

Table 2: Resulting RMSEs for three summer months (June through August) are shown for in-situation predicted and true metrics. Evaluation is given for two commonly used forecast models, gradient boosting regressor (gbr) and time of day temperature (todt), and for the introduced dynamic model selection (dyn) algorithm. Statistics for RMSE are shown for mean, standard deviation (std), minimum (min), median (med), and maximum (max).

	Predicted			True		
	dyn	gbr	todt	dyn	gbr	todt
mean	10.5	10.6	12.5	60.0	67.2	99.6
std	2.7	3.0	2.3	38.5	45.5	127.7
min	5.9	5.9	8.3	17.9	17.9	20.7
med	9.8	9.8	11.7	49.3	61.5	68.6
max	17.8	19.1	17.8	171.5	186.0	686.7

the gradient boosting regressor as baseline, resulting in 1% and 11% respectively.

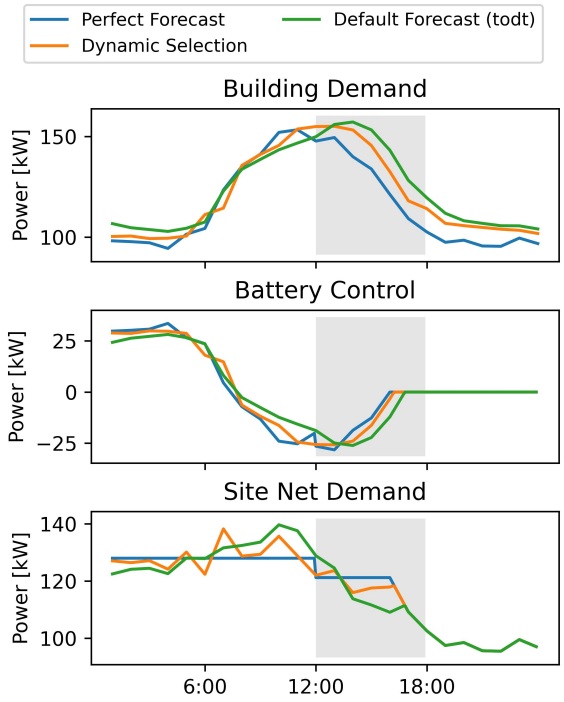

Figure 4: Example application of the dynamic forecast framework to forecast a building demand at Camp Parks military facility and control a battery to reduce electricity cost. The first subplot shows the real building demand (blue), static forecast using the time-of-day-temperature model (green), and the dynamic selection (orange). The second subplot shows the resulting battery dispatch from a model predictive controller using the corresponding forecast model. Positive values indicate battery charging and negative values indicate battery discharging. The last subplot shows the resulting site's net electricity demand, where the computed battery dispatch for each forecast scenario is applied to the real building demand. The period of highest prices and most critical demand to be reduced is shown as shaded gray area from 12:00 to 18:00.

To illustrate the functionality of the dynamic forecast framework, an example application to minimize site electricity cost is given. The example controller utilizes a 24 hour forecast of whole building demand to dispatch a co-located stationary battery. The controller is based on the DOPER[4] model predictive controller (Gehbauer, Lee, and Wang 2023) where a mathematical model of the site is numerically optimized over the 24 hour forecast horizon. The battery is sized based on the peak load demand (i.e., 150 kW and 300 kWh). The site electricity cost is calculated as time-of-use cost for energy and peak demand. The forecast models are re-trained every three days. Figure 4 illustrates the results of the controller over the course of one day for three scenarios. The perfect forecast presents the theoretical baseline where no forecast error is present, which results in an optimally flat site net demand, shown in blue on the last subplot. In this scenario the battery is slowly charged during the cheaper midnight to mid-day hours and then discharged starting from noon, when the highest priced period starts. Battery dispatch is modulated to follow building demand and the resulting flat net demand minimizes peak demand charges. The other two scenarios utilize either the dynamic selection framework, in orange, or a pre-selected default forecast model (i.e., time-of-day-temperature), shown in green. Both of these scenarios show more variability in net load which indicates sub-optimal utilization of demand charges. Extrapolating this one day of operations to a full month, electricity cost would be $ 19,342, $ 19,946 ($ +604), and $ 19,742 ($ +399) respectively for perfect forecast, default forecast, and dynamic selection. The cost effectiveness of the battery is increased by 34%.

[4]Distributed Optimal and Predictive Energy Resources (DOPER): https://github.com/LBNL-ETA/DOPER

Discussion

This paper introduces a new framework which seeks to rapidly reduce user interaction while improving typical forecast performance through a collection of best in class forecast models. The framework was specifically developed for operations at the Camp Park U.S. military facility and includes a number of functionalities which increase both overall model accuracy and forecast reliability. The framework inherently relies on the available models within the forecast model library. While the included eleven models were selected to fit various applications at Camp Parks (i.e., prediction of whole building loads, generation from renewable sources such as photovoltaic, prediction of aggregated whole base net load, etc.) the application to other forecast problems (e.g., building occupancy, single office plug load, electric vehicle usage, etc.) may require future addition of models to the forecast library. However, we believe that the introduction of the new framework and the current selection of forecast models fit many applications related to buildings without the addition of models or any user interactions.

Depending on the specific application, a building control system can have varying level of sensitivity towards forecast errors. The resulting benefit of employing the dynamic forecast framework consequently depends on the specific control system and its reliance on foretasted data. For example, a heuristic control system for HVAC control may seek to forecast internal loads only five minutes ahead and, given the slow dynamics of thermal systems, might be inherently more resilient towards forecast errors. On the other hand, in case of fully predictive control stems, such as model predictive control, forecast error can have drastic effects on performance. The example application given, where a predictive controller uses a 24 hour forecast of a building load at Camp Parks to control a co-located battery to decrease electricity cost, revealed an increased cost effectiveness of 34% when utilizing the dynamic forecast framework.

The objective of this framework is not to provide the ultimate solution for forecasting, but rather to automate a complicated and time consuming process to increase, and in most cases exceed industry standard, forecast accuracy. By autonomously and regularly (e.g., every three days) re-training the library of forecast models with up-to-date data observations, and selecting the best model, the utilization of the best available forecast model is ensured at all times. Manually conducting such retraining and re-iterations every few days would be exceedingly difficult to manage and therefore control designers and building owners typically opt for ”basic” models (i.e., a single model such as time-of-week-temperature) which may provide reasonable forecasts at some time, but are prone to deviate as underlying data evolves. These typical basic models are often developed only once, resulting in higher upfront cost but moderate to no operational cost. In contrast, the introduced forecast selection framework only requires minimal setup, but requires ongoing computing resources to re-train its models. This can be problematic when data exceeds memory thresholds (e.g., greater than 20% of a system's memory - typically 16 GB), or on edge devices which do not provide enough computing power (e.g., embedded systems). The real-time operation of the dynamic forecast framework is ensured through the establishment of two separate asynchronously operating loops, one to infer current forecast generation and one to re-train models, when appropriate. This architecture is designed to offload the computationally intensive re-training process from real-time operation, while also leveraging the multi-core architecture of modern computing hardware.

An additional benefit when actively monitoring forecast model performance is the tracking of expected prediction errors, and detection when errors significantly deviate from expectations. This scheme allows for built-in fault detection and diagnostic to analyze each component's prediction error, and detect impacted systems. Not only can this be used to inform building operators about potential faults, but it can also be used to detect cyberattacks. If any fault or error deviation (either though component failure or malicious attempts) is detected, then the forecast framework would default to a known safe state by either (a) only utilizing models which do not require external inputs such as weather forecast, or (b) load the previously used forecast model and continue producing forecasts but with an additional warning flag that system status is degraded. Continuing to operate even after detection of deviating errors is an important feature for system critical applications such as operations at U.S. military facilities.

The dynamic forecast framework and library will be publicly available as *fcDyn* Python package. Open-sourcing the development allows a greater audience to utilize the development, integrate in manufacturer internal software stacks, or participate in the active development and expand functionality beyond the application in buildings.

Conclusion

The introduced dynamic forecast framework offers accessible state-of-the-art machine learning and regression models for precise weather and demand forecasting, facilitating informed decision-making in energy management. The framework has demonstrated operational effectiveness to capture and adopt to fundamental changes in the underlying data, for example at new installations with little initial data or when site utilization changes. Accurate forecasting through the dynamic model selection is especially important to enhance situational awareness and associated advanced control strategies, as showcased with a 34% increased cost effectiveness of a battery co-located with a building site.

Acknowledgment

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work utilized the ChatGPT 3.5 tool to support the write up of the Introduction section.

References

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45:5–32.

Clark, Jordan D, Brennan D Less, Spencer M Dutton, Iain S Walker, and Max H Sherman. 2019. "Efficacy of occupancy-based smart ventilation control strategies in energy-efficient homes in the United States." *Building and Environment* 156:253–267.

Deetjen, Thomas A, and Inês L Azevedo. 2019. "Reduced-order dispatch model for simulating marginal emissions factors for the United States power sector." *Environmental science & technology* 53 (17): 10506–10513.

Dong, Bing, Vishnu Prakash, Fan Feng, and Zheng O'Neill. 2019. "A review of smart building sensing system for better indoor environment control." *Energy and Buildings* 199:29–46.

Drgoňa, Ján, Javier Arroyo, Iago Cupeiro Figueroa, David Blum, Krzysztof Arendt, Donghun Kim, Enric Perarnau Ollé, Juraj Oravec, Michael Wetter, Draguna L Vrabie, et al. 2020. "All you need to know about model predictive control for buildings." *Annual Reviews in Control* 50:190–232.

Esrafilian-Najafabadi, Mohammad, and Fariborz Haghighat. 2022. "Impact of occupancy prediction models on building HVAC control system performance: Application of machine learning techniques." *Energy and Buildings* 257:111808.

Friedman, Jerome. 2002. "Stochastic Gradient Boosting." *Computational Statistics and Data Analysis* 38 (02): 367–378.

Gehbauer, Christoph, Douglas R. Black, and Peter Grant. 2023. "Advanced control strategies to manage electric vehicle drivetrain battery health for Vehicle-to-X applications." *Applied Energy* 345:121296.

Gehbauer, Christoph, David H Blum, Taoning Wang, and Eleanor S Lee. 2020. "An assessment of the load modifying potential of model predictive controlled dynamic facades within the California context." *Energy and Buildings* 210:109762.

Gehbauer, Christoph, Eleanor S Lee, and Taoning Wang. 2023. "An evaluation of the demand response potential of integrated dynamic window and HVAC systems." *Energy and Buildings* 298:113481.

Geron, Aurelion. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly.

Le, Trang T, Weixuan Fu, and Jason H Moore. 2020. "Scaling tree-based automated machine learning to biomedical big data with a feature set selector." *Bioinformatics* 36 (1): 250–256.

Lee, Eleanor S, Christoph Gehbauer, Brian E Coffey, Andrew McNeil, Michael Stadler, and Chris Marnay. 2015. "Integrated control of dynamic facades and distributed energy resources for energy cost minimization in commercial buildings." *Solar Energy* 122:1384–1397.

Mariano-Hernández, D, L Hernández-Callejo, A Zorita-Lamadrid, O Duque-Pérez, and F Santos García. 2021. "A review of strategies for building energy management system: Model predictive control, demand side management, optimization, and fault detect & diagnosis." *Journal of Building Engineering* 33:101692.

Mathieu, Johanna L, Phillip N Price, Sila Kiliccote, and Mary Ann Piette. 2011. "Quantifying changes in building electricity use, with application to demand response." *IEEE Transactions on Smart Grid* 2 (3): 507–518.

Merabet, Ghezlane Halhoul, Mohamed Essaaidi, Mohamed Ben Haddou, Basheer Qolomany, Junaid Qadir, Muhammad Anan, Ala Al-Fuqaha, Mohamed Riduan Abid, and Driss Benhaddou. 2021. "Intelligent building control systems for thermal comfort and energy-efficiency: A systematic review of artificial intelligence-assisted techniques." *Renewable and Sustainable Energy Reviews* 144:110969.

Montgomery, Douglas C, Elizabeth A Peck, and G Geoffrey Vining. 2021. *Introduction to linear regression analysis*. John Wiley & Sons.

Park, June Young, Mohamed M Ouf, Burak Gunay, Yuzhen Peng, William O'Brien, Mikkel Baun Kjærgaard, and Zoltan Nagy. 2019. "A critical review of field implementations of occupant-centric building controls." *Building and Environment* 165:106351.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Pierre Guerts, Damien Ernst, Louis Wehenkel. 2006. "Extremely randomized trees." *Machine Learning* 63:3–42.

Rumelhart, David, Geoffrey Hinton, and Ronald Williams. 1985. "Learning Internal Representations by Error Propagation." Technical Report, California Univ San Diego La JHolla Inst for Cognitive Science.

Sarker, Iqbal H. 2021. "Machine learning: Algorithms, real-world applications and research directions." *SN computer science* 2 (3): 160.

Schratz, Patrick, Jannes Muenchow, Eugenia Iturritxa, Jakob Richter, and Alexander Brenning. 2019. "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data." *Ecological Modelling* 406:109–120.

Seabold, Skipper, and Josef Perktold. 2010. "Statsmodels: Econometric and statistical modeling with python." *9th Python in Science Conference*.

Szinai, Julia K, Colin JR Sheppard, Nikit Abhyankar, and Anand R Gopal. 2020. "Reduced grid operating costs and renewable energy curtailment with electric vehicle charge management." *Energy Policy* 136:111051.

Tang, Hong, Shengwei Wang, and Hangxin Li. 2021. "Flexibility categorization, sources, capabilities and technologies for energy-flexible and grid-responsive buildings: State-of-the-art and future perspective." *Energy* 219:119598.

Vrettos, Evangelos, and Christoph Gehbauer. 2019. "A Hybrid approach for short-term PV power forecasting in predictive control applications." *2019 IEEE Milan PowerTech*. IEEE, 1–6.

Zhang, Liang, Jin Wen, Yanfei Li, Jianli Chen, Yunyang Ye, Yangyang Fu, and William Livingood. 2021. "A review of machine learning in building load prediction." *Applied Energy* 285:116452.

Appendix

Overview of models in dynamic forecast framework:

- Decision trees: Decision trees are algorithms commonly used for both classification and regression. These supervised learning models predict the values of target variables leveraging boolean True or False statements that arranged in a tree branch and leaf structure. Each decision statement further refines the prediction of the target based on the characteristics of the data until it is categorized into leaves, or groups of data points that are highly similar. For each prediction point the decision tree identifies the leaf with characteristics most similar to the provided data, then returns the classification or mean values of other points in that leaf. (Geron 2019)
  - Random forest regressor (scikit-learn): A meta-estimator that averages a number of decision trees over bootstrapped subsections of the data. (Breiman 2001) The random forest regressor is considered an ensemble model because it utilizes the prediction of many decision trees. This tool creates many different decision trees on a sample of the data set using a randomized subset of the features. These sources of randomization are used to prevent overfitting. Once all the various decision trees are created, the results are averaged to return the final prediction values.
  - Extra trees regressor (scikit-learn): A meta-estimator that averages a number of decision trees over the entire data set. (Pierre Guerts 2006) Extra trees, which is short for extremely randomized trees, adds additional randomness to random forests by using random thresholds when determining which features to use, instead of the most discriminative thresholds as in random forests. This increase in randomization reduces the variance in the model but often slightly increases the bias.
  - Gradient boosing regressor (scikit-learn): Combines multiple decision trees, each trained to resolve the residual error from previous decision trees. (Friedman 2002) Gradient boosting regressor combines decision trees iteratively. The first decision tree fits the data to the best of its ability, but will leave residual errors. Gradient boosting regressor then adds a second decision tree estimating those residual errors, reducing the error from the

first tree. A third tree estimates the residual errors from the second tree, and so on until the regressor has created the specified number of regressors. (Geron 2019)

- Neural networks: Neural networks are machine learning tools providing computational models emulating how biological neurons work in brains. The tools are based on simplified models known as artificial neurons, which leverage one or more binary inputs and a single binary output. The artificial neuron switches between states depending on input values. (Geron 2019)
  - Multilayer perceptron (scikit-learn): A feed-forward neural network which includes at least three layers of fully connected neurons. The first layer is referred to as the input layer, the final is the output layer, and the middle layers are hidden layers. Multilayer perceptrons leverage the backpropagation training algorithm, which is based on gradient descent to efficiently compute every model parameter. (Rumelhart, Hinton, and Williams 1985) Multilayer perceptron models excels at non-linear models or online learning. (Geron 2019)
- Autoregressive: Autoregressive integrated moving average (or ARIMA) models are a type of recurrent network used to evaluate time series data. These models identify trends in the time series data which enable predicting the future over shorter time horizons.
  - SARIMAX (statsmodels): An auto-regressive model based on ARIMA which adds linear seasonal effects and exogenous variable datasets. Typical ARIMA models require users to remove both trend and seasonality impacts (e.g., trend: if the target value is currently increasing by 5% per week; seasonality: people purchase more Christmas decorations in the winter) then add those aspects back into the final prediction. SARIMAX overcomes that limitation by incorporating seasonal and trend prediction algorithms directly into the model. Additionally, most ARIMA models leverage solely time series data and cannot utilise exogenous data sets, such as weather forecasts. The SARIMAX model incldues that capability, increasing the ability of the model to respond to varying scenarios.
- Linear regression : Linear regression models compute the prediction value using a weighted sum of all of the input features, combined with a bias term representing the offset when all inputs are zero. Training consists of identifying the weights for each input feature that minimizes the difference between the training targets and the predicted values. (Geron 2019)
  - Time of day, temperature (in-house): A model that develops a separate linear regression based on outdoor air temperature for each hour of the day. This model uses only the outdoor air temperature as an input feature, and is designed to predict building electricity load as a target. The dataset is split to create subsets for each hour of the day (e.g. midnight to one, one to two, etc), and fits separate instances of linear regression models to each subset. (Mathieu et al. 2011)