A Few Models to Rule Them All:
Aggregating Machine Learning Models

Florian Siepe[1,2], Phillip Wenig[3] and Thorsten Papenbrock[1]

[1]*Philipps University of Marburg, Marburg, Germany*
[2]*Viessmann IT Service GmbH, Allendorf (Eder), Germany*
[3]*Hasso Plattner Institute, University of Potsdam, Potsdam, Germany*

Abstract

Many manufacturers of electrical installations in smart home environments have developed and now offer AI solutions that record and analyze the sensor data from their products. Their goal is to monitor and forecast runtime parameters, such as the energy consumption of heat generators or the cooling performance of air conditioning systems, for predictive maintenance and to optimize the carbon footprint. The training and deployment of such AI models can, though, be costly, necessitating intelligent techniques to consolidate, i.e., aggregate models of individual installations into fewer, but larger models. The aggregation of AI models, however, poses a challenging task due to the complexity of the systems and the variability of (hidden) factors that influence the forecasts. To solve the aggregation challenge, improve the forecasting accuracies and ultimately also reduce the AI deployment costs, this paper explores the concept of consolidating similar machine learning models with a novel clustering approach. We introduce CAML, a novel technique for (C)lustering and (A)ggregating (M)achine (L)earning models with shared characteristics. The clusters effectively capture the unique features of the contained models and can be combined into fewer AI models. Our evaluation shows that the hidden parameters learned by the baseline models are key factors in achieving accurate performance, underlining the importance of these models in the clustering process. Moreover, we demonstrate that by choosing the right model architecture, cluster models offer a higher prediction certainty while exhibiting an only slightly higher average error compared to baseline models. Our experimental results show that CAML outperforms alternative clustering techniques in terms of prediction error and variance across multiple cluster configurations.

Keywords

Machine Learning, Clustering, Model Aggregation, Energy Consumption, Heat Generator

1. Consolidation of AI Model Deployments

In today's digital age, service providers frequently utilize AI and machine learning technologies to analyze the vast swaths of collectable sensor data, offering valuable insights to their customers. A common application is to train individual prediction models for each customer on their specific data. However, this results in a multitude of models that need to be trained, deployed, and maintained, especially in cloud environments, ultimately costing the company a lot of resources and money. In this paper, we investigate this challenge in the context of heat generator

installations and their energy forecasting models.

Accurate energy consumption forecasts are critical for various stakeholders, including utility companies, building managers, and residents, because they enable optimizations in energy usage, predictive maintenance and ultimately cost reductions. The models employed for this purpose use historical sensor data to predict future energy consumption curves. Through careful feature engineering, they are also able to reflect parameter changes (e.g. adjusting the indoor temperature) in their predictions. However, accurate forecasts depend not only on features that are captured by sensors, such as temperatures and power consumptions, but also on various hidden parameters specific to the installation environment, such as the size, age, and insulation type of the building that houses the heating system.

Heating systems usually supply temperature at a constant level, heating water to a specific temperature and circulating it to provide warmth. This approach may, however, not be the most energy-efficient method, particularly during milder weather. Modern heating systems, therefore, use a dynamic approach, adjusting the supply temperature based on the outside temperature measured by an outdoor temperature sensor. This relationship is graphically represented by a heating curve, a crucial part of a heating system's control strategy. These curves can be tailored to the residents' preferences, which is an important variable for the accuracy of energy consumption forecasts. Additional factors include the desired indoor temperature, which is set via a thermostat, and the application of a night setback concept. The latter lowers the temperature during unoccupied periods like nighttime to save energy. These strategies make modern heating systems usually more efficient, but the interplay of these novel features combined with the hidden housing parameters of the installations make it difficult for manufacturers to develop generalizable AI models, such as the global forecasting model shown in Figure 1a. Instead, the current practice is to train, deploy and maintain a forecasting model (with potentially different hyper-parameters and model architectures) for every installation, as show in Figure 1b. Although the predictions of these models are decently accurate, their deployment and maintenance is expensive.

Despite the complex feature interactions and the numerous hidden parameters, many installations and, hence, their AI models are sufficiently similar, such that, as depicted in Figure 1c, a smaller set of AI models could manage their forecasts. To find these partially generalizable models, we propose to cluster similar heat generator models and consolidate every cluster into only one cluster model. These cluster models are expected to encapsulates the unique features and encoded knowledge of each cluster. In our setting, this encoded knowledge refers to constant parameters that are relevant in the learning process but not directly visible within the data set, such as the size or insulation of buildings. By consolidating n individual models into m cluster models with an effective clustering, the number of models necessary for deployment and serving can be reduced significantly. To find these clusterings, we evaluate three different approaches: a) clustering the models by their underlying time series *training data*; b) clustering the models by their *output data*, which is their predictions; and c) clustering the models by their *cross performance*, which is the performance on each others training data. We demonstrate that approach c) delivers the most promising results, making it our suggested solution.

More specifically, we introduce CAML, a novel technique for (C)lustering and (A)ggregating (M)achine (L)earning models of any type. CAML uses a custom *cross performance* similarity function and hierarchical clustering to find clusters of AI models with similar hidden features;
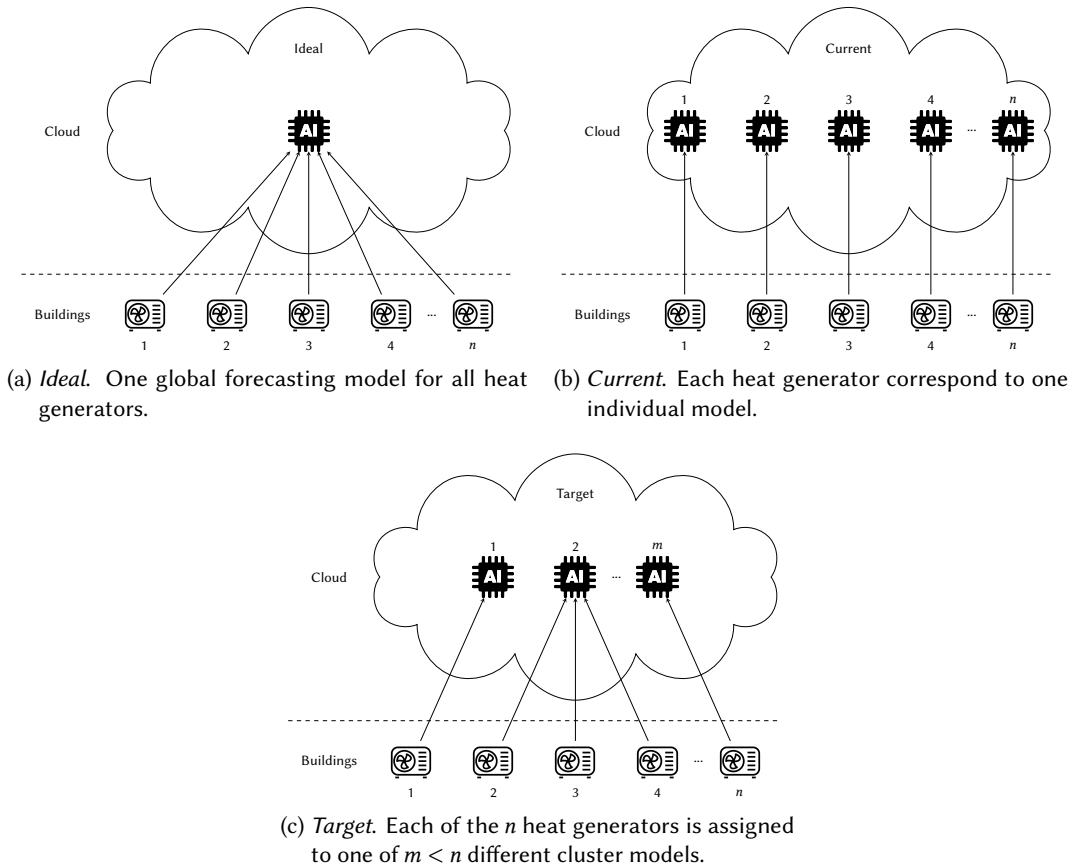
(a) *Ideal.* One global forecasting model for all heat generators.

(b) *Current.* Each heat generator correspond to one individual model.

(c) *Target.* Each of the n heat generators is assigned to one of $m < n$ different cluster models.

Figure 1: Three types of deployment architectures: The theoretically ideal global model approach (1a), the current individual models approach (1b), and the target cluster model approach (1c). Heat generator installations send their sensor recordings into the cloud and receive temperature and energy forecasts.

it then trains new *cluster models* on the clusters' training data to consolidate the member models into one representative model. Our evaluation demonstrates that cross performance is more effective than input- and output-based model similarities; it also shows that the cluster models are significantly more accurate than a global model and almost as accurate as the many individual models. Hence, our key contributions are as follows:

1. **Cross performance**: We propose an effective similarity measure for model clustering that compares two models by their pair-wise test accuracies.
2. **Hierarchical clustering**: We introduce an unsupervised, hierarchical clustering approach with Ward-linkage based on the cross performance similarity measure that grands the ability to configure the number of target clusters, i.e., the trade-off between deployment costs and forecasting accuracy.
3. **CAML**: We present an algorithm that automates the entire model consolidation pipeline, including preprocessing, matrix computation, clustering, and subsequent retraining.

2. Related Work

Clustering is an unsupervised machine learning technique that partitions objects into (mutually exclusive) partitions based on their similarity w.r.t. a specific similarity measure. Clustering algorithms can be categorized into several types. These include *partitioning methods* that divide the data set into a pre-set number of clusters (e.g. K-Means [1]), *hierarchical methods* that aim to create a cluster linkage tree that models the relationships in the data best (e.g. Bottom-Up [2]); and *density-based methods* that recognize clusters as high-density regions of objects separated by regions of low density (e.g. Meanshift [3] or DBSCAN [4]). Most of the mentioned standard clustering algorithms either rely on euclidean distance as a fixed distance metric (e.g., K-Means), which makes them rather inflexible, or the number of desired clusters cannot be directly specified (e.g., Meanshift, DBSCAN). In our work, we therefore focus on hierarchical clusters, which can handle any given distance metric and desired number of clusters.

Time Series Clustering algorithms can be used to cluster AI models by their input and/or output data. Bandara et al., for example, use selected features extracted from time series data, such as trend or seasonality, to apply standard clustering algorithms [5]. A similar approach is employed by Räsänen and Kolehmainen, who focus specifically on energy consumption data [6]. Algorithms for time series clustering in general rely on time series distance measures, such as Dynamic Time Warping (DTW) [7], Move-Split-Merge (MSM) [8], or Time-Warp-Edit (TWE) [9]. Paparrizos and Gravano presented k-Shape, an algorithm for clustering univariate time series, which has proven to be effective in several works [10, 11, 12]. We therefore chose shape-based distance (SBD) in combination with hierarchical clustering for our reference clustering approaches that are based on the input time series similarities (see approach a)).

Model Consolidation describes the process of combining several individual models into a single model. Bakker and Heskes demonstrated that in many cases a smaller set of representative models can adequately summarize an ensemble of neural network models [13]. In their approach, they leveraged the outputs of these models on a static data set for cluster assignment followed by an optimization step for finding the model as a cluster representative that minimizes the average cluster distance between the models of the cluster and its center. This approach is similar to our baseline approach b), which also relies on the models' outputs but instead uses the outputs as direct embedding within the clustering procedure. A common technique for aggregating models is ensemble learning. Sarkar et al. discuss the use of different ensemble learning models for short-term electric load forecasting. The authors used real-time load data and meteorological parameters for data analysis [14]. Khan et al. presents a spatial and temporal forecasting model ensemble of LSTM and GRU deep neural networks for short-term electric consumption forecasting [15]. Another common technique is response-based model distillation as formulated by Hinton et al., which summarizes a large but precise teacher model within a smaller student model [16]. The proposed single-teacher distillation can be extended to a multi-teacher distillation for ensemble learning by either aggregating the teachers' responses or by weighting the responses [17]. In summary, most of the mentioned works imply training large, global models. Our experiments will show that in our specific domain, training an accurate, global teacher models is not possible, which eliminates model distillation as an option for our model consolidation use case. Using model distillation on multiple global models is possible but

typically impractical and expensive due to high computational and time demands. This paper mainly discusses clustering and aggregating pre-trained models based to form new consolidated models, a strategy distinct from federated learning which centers on training a single model across many decentralized nodes.

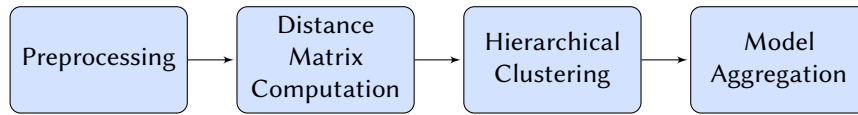## 3. CAML - Clustering and Aggregating Machine Learning Models

Figure 2: Architecture of CAML with its four mayor steps: preprocessing, distance matrix computation, hierarchical clustering, and model aggregation.

This section introduces CAML, our proposal for a (C)lustering and (A)ggregation technique for (M)achine (L)earning models. As visualized in Figure 2, CAML comprises four steps, which are *data preprocessing*, *distance matrix computation*, *hierarchical clustering*, and *model aggregation*. To execute these steps, CAML requires only one user-defined parameter: the number of clusters k that controls the granularity of the clustering. The selection of k strongly depends on the desired use case. Most commonly, k represents the number of cluster models the user wishes to deploy. However, an exhaustive search of k can be performed to meet certain prediction accuracy goals of the resulting cluster models compared to the individual models. Because CAML embodies hierarchical clustering, which produces a linkage matrix, multiple different clusterings can be extracted without recomputing the actual clustering procedure. We now introduce the four steps of CAML in more detail.

Preprocessing: Time series that contain anomalous data points (e.g., due to faulty sensor readings) are harmful to any clustering attempt. The removal of outliers is an essential pre-processing step as it helps to reduce the influence of anomalous data points on subsequent steps of clustering and model aggregation. These outliers, if not addressed, can distort overall patterns and relationships within the data, leading to suboptimal clustering results. Our CAML algorithm, therefore, initially preprocesses the data on which the individual models have been trained and tested, replacing the outlier data points in the time series with the value of the next, non-anomalous data point in the series. In this context, we define an outlier as a data point where at least one of its feature values is greater or less than v standard deviations from the mean of that feature. By default, we choose v to be 3, because we expect that each feature x follows approximately a gaussian distribution, i.e., $P(\mu - 3\sigma \leq x \leq \mu + 3\sigma) \approx 0.9973$. Thus, only the extreme outliers are removed, and most of the original data remains.

Next, the time series of each model is split into training and test sets. We denote the training sets of model m_i as $T_i = (X_i, Y_i)$ and the test sets as $t_i = (x_i, y_i)$, with X_i and x_i as features and Y_i and y_n as labels of the model, respectively. It is important to note that these splits need to be the same as used in the training procedure of the baseline model. The test sets are used in our

proposed distance function for computing a complete distance matrix of the models.

Distance Matrix Computation: The hierarchical clustering step of CAML is based on a distance matrix that stores all pair-wise distances between baseline models. These pair-wise distances should measure the dissimilarity between two machine learning models m_i and m_j and are computed with a custom distance function, which is based on the pair-wise cross-validation of the models on each other's test sets. In other words, the distance between model m_i and m_j is the mean of m_i's loss on m_j's test data and m_j's loss on m_i's test data. We calculate this loss as the Mean Absolute Error (MAE) on the respective test sets. The intuition behind this distance function is to measure how well m_i performs in the specific environment of m_j and vice versa. If the loss of m_i in the setting of m_j is close to m_j's loss in its environment, m_j can be replaced with m_i. By performing this calculation bidirectionally, we measure the extent to which the two models can replace each other.

Formally, let $m : R^{u \times v} \to R^v$ be a prediction model, which maps v data points with u different features to v predictions, $l : R^v \times R^v \to \mathbb{R}_0^+$ be a non-negative loss function (e.g., MAE) and $t_i = (x_i, y_i)$ be the test set of the features x_i and the label y_i of a model m_i. The distance between any model m_i and m_j can be computed as:

$$d(m_i, m_j) = \frac{1}{2} \left(l(m_i(x_j), y_j) + l(m_j(x_i), y_i) \right) \tag{1}$$

Here, $m_i(x_j)$ denotes the predicted output of model m_i on the test set of model m_j, and $l(m_i(x_j), y_j)$ is the loss between the predicted output $m_i(x_j)$ and the true label y_j. The distance function is symmetric, i.e., $d(m_i, m_j) = d(m_j, m_i)$, and it is non-negative, i.e., $d(m_i, m_j) \geq 0$.

Hierarchical clustering: For the actual clustering of the AI models, CAML uses a hierarchical clustering approach [2] that can create any number of k pre-specified clusters with sophisticated linkage methods. Hierarchical clustering successively merges clusters of objects into ever larger clusters with increasing merge distance, which effectively creates a cluster tree. Given the distance matrix from the previous step, the hierarchical clustering algorithm proceeds as follows:

1. *Initialize:* Each object, i.e., AI model is considered as one initial, separate cluster.
2. *Merge:* The two clusters with the smallest distance to each other with respect to the linkage method and distance function are merged into a new cluster.
3. *Iterate:* Unless all the clusters have been merged into one cluster, re-iterate with step 2.

The linkage method defines the distance of any pair of clusters based on the pair-wise object distances. The distances then determine the most similar pair of clusters in each iteration. Because the choice of the linkage method can significantly influence the results of the hierarchical clustering, we considered various popular methods including *single linkage*, *complete linkage*, and *average linkage*, which correspond to the minimum, maximum and average distances of objects in the two compared clusters, and found *Ward linkage* to be the most effective method in this step.

The Ward linkage method [18] uses the within-cluster variance, which is the variance of of all pair-wise object distances within some cluster, for the distance calculation: The distance of two clusters is the *increase in total within-cluster variance* of the merged cluster w.r.t. its

two base clusters. We choose Ward, because the pair-wise distances between models within a cluster should be possibly low, such that the models can replace each other. In mathematical terms, if we denote c_i and c_j as two clusters and c_{ij} as the resulting cluster after merging c_i and c_j, then the total increase in squared distance due to merging, known as Ward's criterion, is:

$$\Delta\sigma^2 = \sum_{x \in c_{ij}} d(x, \mu(c_{ij}))^2 - \sum_{x \in c_i} d(x, \mu(c_i))^2 - \sum_{x \in c_j} d(x, \mu(c_j))^2 \text{ and } \mu(c) = \arg\min_{i \in c} \sum_{j \in c} d(i,j) \quad (2)$$

where $\mu(c)$ is the centroid of the observations in cluster c, d is a distance function based on the distance matrix, and x are the data points.

Model aggregation: The model aggregation takes as input the clustering hierarchy and produces as output a set of k *cluster models*. The cluster hierarchy is stored as a linkage matrix and can be visualized as a dendrogram. At first, the aggregation process simply cuts the dendrogram at the specific depth that creates k disjoint clusters. Then, it consolidates every cluster into a single cluster model by training a new model on the combined training data of all individual models of that cluster. More specifically, let $C = \{C_1, \dots, C_k\}$ be any clustering of baseline models. The cluster model M_i for a given cluster C_i with size $n = |C_i|$, the models $m_{i,1}, \dots, m_{i,n}$ and their respective test sets $t_{i,1}, \dots, t_{i,n}$, where $t_{i,j} = (x_{i,j}, y_{i,j})$, and training sets $T_{i,1}, \dots, T_{i,n}$, where $T_{i,j} = (X_{i,j}, Y_{i,j})$, along with a fitting function f, is computed as in Equation 3 by merging all data of the cluster.

$$M_i = f(\bigcup_{j=1}^{n} X_{i,j}, \bigcup_{j=1}^{n} Y_{i,j}) \quad (3)$$

The AI models within a cluster are similar in their behavior, but they might use very different types of models (e.g. linear regression, random forest regression, LSTMs etc.) and hyperparamter settings. Our clustering results in fact show that model architectures have little impact on the clustering results. The consolidation, therefore, needs to find a model architecture with similar or higher capacity than the baseline models to capture all individual properties. Because the best model for consolidation depends on the concrete baseline models, the model aggregation needs to test a set of diverse model types to find the most effective model architecture for the cluster models. Within our implementation, we test five models, which are *ExtraTreesRegressor* [19], *LightGBM* [20], *XGBoost* [21], *N-Beats* [22], and randomly selecting one baseline model to serve as a cluster model. Comparing the model performances can easily been done on the validation data sets of the respective clusters. We present more in-depth results in Section 5.

4. Metrics for Evaluating the Model Aggregation Effectiveness

In this section, we describe metrics for measuring the effectiveness of our clustering-based model aggregation. With CAML, we trained new cluster models that should replace their baseline models. To measure how well a cluster model generalizes its baseline models, we need a metric, with which we can compare the effectiveness of different cluster model sets. To evaluate the effectiveness, we measure the accuracy of each cluster model and compare it to the unconsolidated accuracies of the respective baseline models. The performance of a cluster model is then the average decrease in loss of the trained cluster model compared to all of its

baseline models. Both the cluster model and its baseline models are tested with the respective test sets of the corresponding baseline models.

First of all, we measure the accuracy of a set of predictions $\hat{y}$ w.r.t. the correct test labels y as the *mean average error* (MAE) loss:

$$\text{MAE}(\hat{y}, y) = \frac{1}{|y|} \sum_{i=1}^{|y|} |y_i - \hat{y}_i| \tag{4}$$

Given a cluster model M_i for cluster C_i with $1 \leq i \leq k$ and cluster size $|C_i|$, we calculate the effectiveness score $\mu_c(i)$ of M_i on cluster i as the mean MAE loss over all test sets $t_{i,j} = (x_{i,j}, y_{i,j})$ whose model $m_{i,j}$ belongs to cluster i (Equation 5). Additionally, we compute an effectiveness score $\mu_b(i)$ of all baseline models $m_{i,j}$ of cluster i for comparison. Here, we compute the mean MAE loss over all baseline models $m_{i,j}$ in C_i on their own test sets (Equation 6).

$$\mu_c(i) = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \text{MAE}(M_i(x_{i,j}), y_{i,j}) \quad (5) \qquad \mu_b(i) = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \text{MAE}(m_j(x_{i,j}), y_{i,j}) \quad (6)$$

Let $N = \sum_{i=1}^{k} |C_i|$ be the number of all baseline models, we can now calculate the overall accuracy of all cluster models and all baseline models over all clusters and by aggregating the individual scores $\mu_c(i)$ and $\mu_b(i)$ to total scores μ_C and μ_B, respectively. For this, we apply the weighted mean with the cluster size $|C_i|$ as weight (Equations 7 and 8).

$$\mu_C = \frac{1}{N} \sum_{i=1}^{k} |C_i| \cdot \mu_c(i) \qquad (7) \qquad \mu_B = \frac{1}{N} \sum_{i=1}^{k} |C_i| \cdot \mu_b(i) \qquad (8)$$

To judge the overall effectiveness of the clustering-based model aggregation we can simply compare μ_C to μ_B or some $\mu_{C'}$. If $\mu_C < \mu_B$ for some clustering C, the cluster models performed better than the baseline models. The value of $\mu_C - \mu_B$ can be interpreted as the additional loss one encounters when aggregating the baseline models. We also consider the spread of the clustering scores as the variance σ_c^2 of $\mu_c(i)$ and σ_b^2 of $\mu_b(i)$ (Equations 9 and 10). For σ_B^2 we consider variance of the baseline model m_js' prediction error on its test set $t_j = (x_j, y_j)$.

$$\sigma_C^2 = \frac{1}{N} \sum_{i=1}^{k} |C_i| \cdot (\mu_c(i) - \mu_C)^2 \quad (9) \qquad \sigma_B^2 = \frac{1}{N} \sum_{j=1}^{N} (\text{MAE}(m_j(x_j), y_j) - \mu_B)^2 \quad (10)$$

Because the model consolidation effectiveness depends on the clustering effectiveness and the aggregation effectiveness, we evaluate both aspects in Section 5.

5. Experiments

In this section, we first outline our experimental setup (Section 5.1) and, then, discuss our experimental results with CAML (Section 5.2) to demonstrate the algorithms effectiveness.

5.1. Experimental Setup

Measuring Quality: To assess the quality of our clustering-based aggregation technique, we leverage μ_C of the cluster models and μ_B of the baseline models. The baseline models act as a lower bound for the MAE, because they fit the individual setups and their (hidden) hyperparameters best. As an upper bound for the MAE, we train a global model (with different architectures) using the entire training data set without any clustering. To evaluate the proposed aggregation technique, we also test a clustering-based approach that randomly selects one of the baseline models in every cluster as a reprepsentative.

Benchmark Approaches: To evaluate the proposed *cross performance*-based clustering, we benchmark two additional clustering approaches: *training data*-based and *output data*-based. The training data-based approach clusters the data that the models have been trained on. Here, we use hierarchical clustering with Ward linkage with the Shape-Based Distance (SBD) derived from the k-shape algorithm [10, 11]. Because SBD is designed for univariate time series, we apply it on each feature of our multivariate time series individually and sum the distances. The output data-based approach employs the models and their outputs. We consider all training data as one big set and randomly select p individual measurements from this set and add gaussian noise to them to prevent overfitting and increase robustness. These data points then serve as input for the models to compute a signature for each model, represented as p-dimensional vectors of the models' output. Then, hierarchical clustering - this time using euclidean distance - is performed on these signatures.

Technology: CAML is written in Python 3.10 and can handle any model that implements the Scikit-learn [23] regressor interface. For hierarchical clustering with Ward [18], we use the implementation from Scipy [24]. The cluster models are built using the time series forecasting library Darts [25]. In particular, we evaluate the *ExtraTreesRegressor* [19], *LightGBM* [20], *XGBoost* [21], and *N-Beats* [22]. The hyper-parameters of both, cluster and global models, have been tuned using Optuna [26]. The source code for CAML can be found on GitHub[1].

Data set: To assess CAML's performance, we ran the algorithm on a real-world data set from our industry partner. The data set comprises the models and time series of 370 heat pumps. The multivariate time series vary in length, ranging from 200 to 607 time steps with daily recordings of the energy consumption of these heat pumps, along with their associated measurements (e.g., outdoor temperature, supply temperature, date, etc.), which serve as input features for the models. The baseline models are built using Scikit-learn [23] and its implementations of the *ExtraTreesRegressor* [19] and *GradientBoostingRegressor* [27].
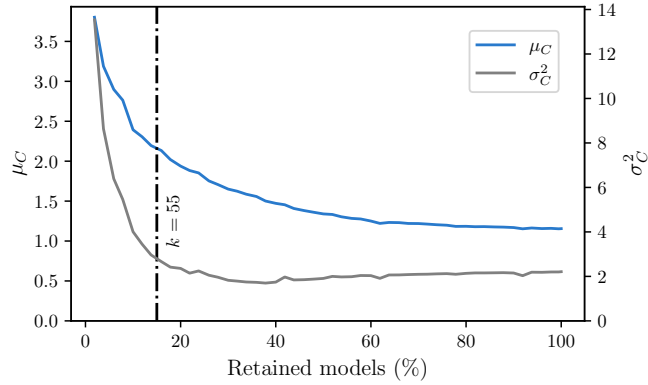
---

[1] https://github.com/floriansiepe/CAML

Figure 3: Evaluation of the number of target clusters k. The clustering scores μ_C and σ_C^2 over the share of retained models in percent with a selected clustering of $k = 55$. In total, the data set consists of 370 models.

5.2. Experimental Results

Because CAML creates a hierarchical clustering, the algorithm needs to choose a number of clusters k, which is the number of to-be-created cluster. Our first experiment, therefore, evaluates the influence of k on the clustering scores. The results are shown in Figure 3. The x-axis shows the share of retained models in percent, while the y-axis shows the scores μ_C and σ_C^2; the more clusters we use, the better each aggregated model can specialize on the specific installations. The goal is to create as few clusters as possible, while keeping the clusters' mean MAE μ_C acceptably small. The depicted curve is an effective tool to tune k for a specific application. For our application with 370 models, we chose $k = 55$ (see vertical line) with the *elbow method* [28] in combination with μ_C as supervised metric of the cluster models prediction error instead of the usually employed sum of squared errors. Also practical constraints on the number of deployable models from our industry partner influence the choice of k.

In a second experiment, we evaluate the three clustering approaches based on (a) *training data*, (b) *output data*, and (c) *cross performance* (CAML) by their MAE scores. For this experiment, the cluster models are built using the original model architectures, which are either *ExtraTreesRegressor* or *GradientBoostingRegressor*, depending on which yields better accuracy; we evaluate the model selection for the cluster models in the next experiment. Figure 4 plots the measured MAE scores for the three clustering approaches. Each subfigure plots the average MAE $\mu_b(i)$ of the baseline models on their respective test set in each cluster (x-axis) against the average MAE $\mu_c(i)$ of the cluster models on all test sets in their cluster (y-axis). If a measurement point is above the diagonal line, the baseline models performed better in this cluster; if a point is below the diagonal line, the cluster model performed better. The size of the marker indicates the cluster size. For every subfigure, we also provide the scores of the entire clustering (see 4) for the baseline models (μ_B and σ_B^2) and the cluster models (μ_C and σ_C^2). The measurements in Figure 4 show that *cross performance*-based clustering (with CAML) results in a lower prediction error and variance than *training data* or *output data*-based clustering. The *training data*-based clustering performs worst, because the hidden features are not represented in the input data

and, therefore, do not influence this clustering – if they were represented, a globally aggregated model could learn from them as well and the clustering would not have been necessary to begin with. The *output data* and *cross performance*-based clustering approaches, however, both work well for the model clustering task, because they both capture the hidden features. The *cross performance*-based approach, however, has a slightly lower error and more dense clustering with less variance than the *output data*-based approach. For this reason, we propose the former with CAML.

In a third experiment, we evaluate CAML's model aggregation. More specifically, given a specific clustering, we evaluate different model architectures and how models within a cluster can be aggregated. For this, we choose the clustering of CAML with $k = 55$ (Figure 4) and implement multiple model architectures, both as cluster models and as global models. Figure 5 shows the distribution of the MAEs of these model architectures in comparison to the baseline models. Considering only the mean performances, we see that the individual models, i.e., the baseline models perform best, the cluster models sacrificed a little bit of precision for having a much smaller overall number of models, and training only one global model provides the worst performance. The fact that the cluster models' mean MAE is very close to the baseline's mean MAE demonstrates the effectiveness of CAML's clustering. The *ExtraTrees* and *LightGBM* cluster models even outperform the baseline models in terms of MAE variance, i.e., prediction certainty. Another observation that highlights the effectiveness of the proposed clustering is that the performance of a randomly selected baseline model from each cluster as cluster model offers a similar – although slightly worse – performance than all other cluster models, which retrain on the clusters' data set. Considering the different model architectures, every global model performs worse than its respective cluster model counterpart. This also shows that the clustering effectively captures some hidden features. The model architectures, however, profit differently from the clustering. Most notably, the neuronal network-based *N-Beats* model hardly improves with the clustering, because it tends to overfit the training data and, hence, has the same problems on clustered and non-clustered inputs; both the cluster and global *N-Beats* model perform worse than all cluster models. In summary, *ExtraTrees*, *LightGBM*, and *XGBoost* all
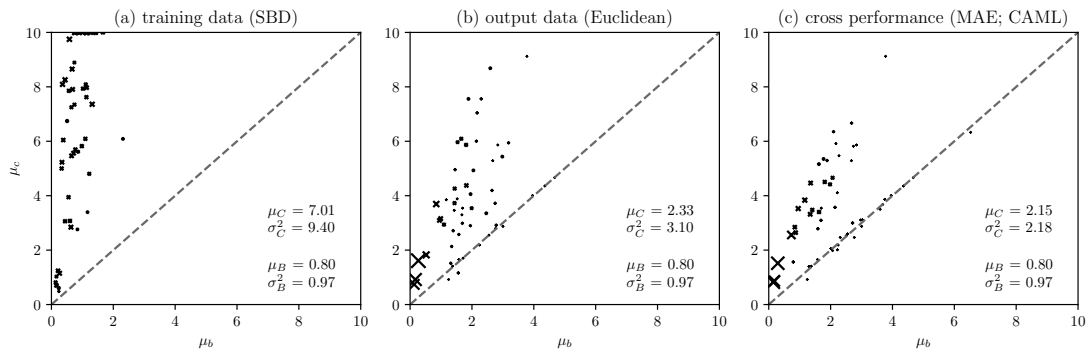

Figure 4: Evaluation of clustering approaches. The MAEs of the clustering approaches based on (a) *training data*, (b) *output data*, and (c) *cross performance* (CAML) measured as μ_C and compared to the MAE of the baseline models μ_B. The marker size corresponds to the size of each cluster.
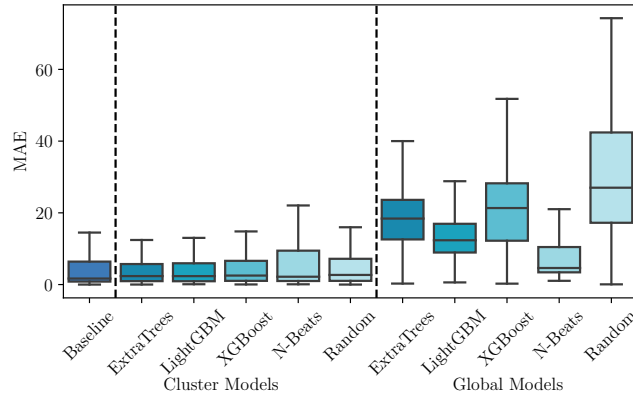
Figure 5: Evaluation of cluster models and overall success. The MAE distribution of the baseline models and different model architectures for cluster and global models.

performed very well as cluster models for the aggregation step.

6. Conclusion

In this paper, we introduced CAML, a novel clustering-based aggregation technique for machine learning models that have been trained in specific environments with certain hidden but constant features. Due to these hidden features, no global model can replace all individual models. The clusters of models that CAML creates effectively capture these hidden features and serve to consolidate their models into much fewer models with still very good precision. With the proposed hierarchical clustering approach, data scientists can tune the trade-off between the number of to-be-deployed models and the models' precision. Our experimental results demonstrate that CAML outperforms all globally aggregated models as well as our benchmark approaches, which cluster the models using their training data and output data, in terms of prediction error and variance. The consolidated cluster models tend to have a slightly higher average error than the baseline models, but the proposed model architectures have an overall lower variance, i.e., prediction certainty than the baseline models. In the context of the investigated application, CAML offers a solution for the aggregation of energy consumption models, enhancing the practical value of these models for utility companies, building managers, and consumers, while simultaneously reducing operational costs.

Future Work: We evaluated CAML on a data set with only 370 models. Due to the positive results obtained with the proposed consolidation, the next step is to deploy CAML on a data set with, at the time of writing, several hundred-thousand models and their respective data. To make this possible, further research on enhancing CAML's performance and/or scalability is needed, given that CAML requires the computation of a distance matrix with an expensive distance function and a runtime complexity of $\mathcal{O}(n^2)$.

References

[1] J. MacQueen, Classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

[2] B. Everitt, S. Landau, M. Leese, D. Stahl, Cluster Analysis, 5 ed., John Wiley & Sons, 2011.

[3] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 603–619. doi:10.1109/34.1000236.

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A densitybased algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, Kdd'96, AAAI Press, Portland, Oregon, 1996, pp. 226–231.

[5] K. Bandara, C. Bergmeir, S. Smyl, Forecasting across time series databases using long short-term memory networks on groups of similar series, CoRR abs/1710.03222 (2017). URL: http://arxiv.org/abs/1710.03222. arXiv:1710.03222.

[6] T. Räsänen, M. Kolehmainen, Feature-based clustering for electricity use time series data, in: M. Kolehmainen, P. J. Toivanen, B. Beliczynski (Eds.), Adaptive and Natural Computing Algorithms, 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers, volume 5495 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 401–412. URL: https://doi.org/10.1007/978-3-642-04921-7%5F41. doi:10.1007/978-3-642-04921-7_{4}{1}.

[7] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing (1978). doi:10.1016/b978-0-08-051584-7.50016-4.

[8] A. Stefan, V. Athitsos, G. Das, The move-split-merge metric for time series, IEEE Transactions on Knowledge and Data Engineering (TKDE) 25 (2013) 1425–1438. doi:10.1109/tkde.2012.88.

[9] P.-F. Marteau, Time warp edit distance with stiffness adjustment for time series matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 306–318. doi:10.1109/tpami.2008.76. arXiv:cs/0703033.

[10] J. Paparrizos, L. Gravano, k-shape: Efficient and accurate clustering of time series, in: Proceedings of the International Conference on Management of Data (SIGMOD), 2015. doi:10.1145/2949741.2949758.

[11] J. Paparrizos, L. Gravano, Fast and accurate time-series clustering, ACM Transactions on Database Systems (TODS) 42 (2017). doi:10.1145/3044711.

[12] J. Yang, C. Ning, C. Deb, F. Zhang, D. Cheong, S. E. Lee, C. Sekhar, K. W. Tham, k-shape clustering algorithm for building energy usage patterns analysis and forecasting model accuracy improvement, Energy and Buildings 146 (2017) 27–37. URL: https://www.sciencedirect.com/science/article/pii/S0378778817305352. doi:10.1016/j.enbuild.2017.03.071.

[13] B. Bakker, T. Heskes, Clustering ensembles of neural network models, Neural Networks 16 (2003) 261–269. URL: https://www.sciencedirect.com/science/article/pii/S0893608002001879. doi:10.1016/s0893-6080(02)00187-9.

[14] D. Sarkar, T. Ao, S. K. Gunturi, Bootstrap aggregating approach to short-term load forecasting using meteorological parameters for demand side management in the north-eastern region of india, Theoretical and Applied Climatology 148 (2022) 1111–1125. doi:10.1007/s00704-022-03933-9.

[15] A.-N. Khan, N. Iqbal, A. Rizwan, R. Ahmad, D.-H. Kim, An ensemble energy consumption forecasting model based on spatial-temporal clustering analysis in residential buildings, Energies 14 (2021) 3020. doi:10.3390/en14113020.

[16] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, ArXiv.org (2015). doi:10.48550/arxiv.1503.02531. arXiv:1503.02531.

[17] H. Zhang, D. Chen, C. Wang, Confidence-aware multi-teacher knowledge distillation, in: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2022, pp. 4498–4502. doi:10.1109/icassp43922.2022.9747534.

[18] J. H. Ward, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58 (1963) 236–244. URL: http://www.jstor.org/stable/2282967.

[19] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (2006) 3–42.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper%5Ffiles/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[21] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Kdd '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785. arXiv:1603.02754.

[22] B. N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, N-beats: Neural basis expansion analysis for interpretable time series forecasting, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=r1ecqn4YwB.

[23] F. Pedregosa, 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, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[24] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[25] J. Herzen, F. L. A¤ssig, S. G. Piazzetta, T. Neuer, L. A©o Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. KoÅ›cisz, D. Bader, F. A©dÃ©rick Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, G. A«l Grosch, Darts: User-friendly modern machine learning for time series, Journal of Machine Learning Research

23 (2022) 1–6. URL: http://jmlr.org/papers/v23/21-1177.html.

[26] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperpa-rameter optimization framework, in: Proceedings of the International Conference on Knowledge discovery and data mining (SIGKDD), ????

[27] J. H. Friedman, Greedy function approximation: A gradient boosting machine., The Annals of Statistics 29 (2001) 1189–1232. doi:10.1214/aos/1013203451.

[28] R. L. Thorndike, Who belongs in the family? 18 (1953) 267–276. doi:10.1007/bf02289263.