ModelForge: A Metric Approach to Machine Learning Model Consolidation

Florian Siepe
Viessmann IT Service GmbH
Allendorf (Eder), Germany
0009-0008-5911-5327

Thomas Peter Prinz
BMW AG
Munich, Germany
0009-0002-8170-1097

Thorsten Papenbrock
Philipps University Marburg
Marburg, Germany
0000-0002-4019-8221

***Abstract*—Many companies provide their customers with digital services for analytical purposes that are backed by modern expert machine learning models specifically trained for individual appliances. These models are often easy to train, but the deployment and operation of numerous individual machine learning models is a resource-intensive challenge. Due to hidden features in the individual appliances, consolidating all expert models into one model is often not possible. However, certain groups of models with similar appliances usually can be combined without (significant) loss in performance. To find these groups without knowledge of the actual hidden features, this paper proposes the consolidation algorithm *ModelForge*, which is based on a novel embedding strategy for model clustering. The *Prediction Loss* strategy embeds arbitrary models into Euclidean space in a way that close models share similar properties and can, therefore, effectively be consolidated. We validate *ModelForge* across four diverse domains, which are energy forecasting, timeseries anomaly detection, weather postprocessing, and house price prediction, to show that it yields more accurately consolidated models than previous works and alternative embedding strategies.**

***Index Terms*—machine learning, clustering, model consolidation**

I. Introduction

In the rapidly evolving landscape of digital services, artificial intelligence (AI) and machine learning (ML) technologies play a crucial role in analyzing vast quantities of sensor data and providing valuable insights to customers. A common challenge in this domain is the resource-intensive requirement of deploying and operating vast amounts of expert models for individual customers and their specific appliances. An energy consumption forecasting service for heating installations, for example, provides value by enabling energy optimizations, predictive maintenance, and cost reduction. Accurate energy forecasts, however, depend on numerous, appliance-specific and usually hidden, but constant factors, such as the building size, insulation type, heating strategies, and building location. While these constants are learned by the individual expert model they are not explicitly available.

To partially consolidate expert models with similar hidden properties, previous work proposed CAML (Clustering and Aggregating Machine Learning models) [1], a model clustering framework that utilizes *Cross Performance* as a distance function to quantify model similarity. The distance function measures how accurately two models perform on each other's data and creates a distance matrix for all models from these measurements. While being effective for capturing functional similarity, this approach suffers from two significant limitations: Since it violates the triangle inequality, it lacks mathematical metricity and, because of the construction of a full distance matrix, it has quadratic runtime complexity.

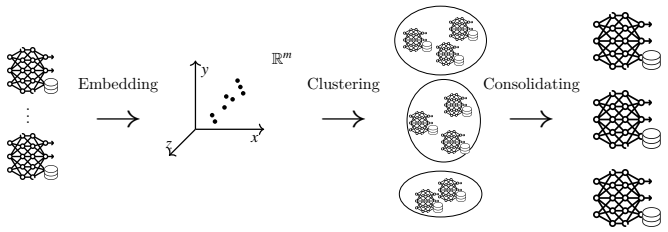

Fig. 1. Workflow of *ModelForge* consisting of the three sequential stages *Embedding*, *Clustering* and *Consolidating*.

Building upon the results of CAML, this paper presents enhanced methodologies for model clustering and aggregation that specifically address the discussed limitations with the help of a novel embedding strategy. Our proposed framework *ModelForge*, which is depicted in Fig. 1, embeds machine learning models into Euclidean space based on their predictions on a carefully constructed reference dataset, clusters similar models, and creates consolidated representatives. This approach better captures the nuanced relationships between models with varying underlying parameters while at the same time also enhancing the scalability of the representation construction by reducing computational complexity from $\mathcal{O}(n^2)$ for computing a distance matrix to $\mathcal{O}(n)$ for generating embeddings.

In our evaluation, we consider four use cases from diverse domains: energy forecasting of heat generators [1], timeseries anomaly detection [2]–[5], probabilistic weather forecast postprocessing [6] and house price prediction [7]. The datasets of these use cases incorporate varying hidden characteristics, regional climate patterns and price

This work was supported by the Viessmann IT-Service GmbH.

markets as well as varying patterns within timeseries, for a meaningful assessment of *ModelForge*'s generalizability. More specifically, our contributions are the following:

- *Prediction Loss:* We propose an effective embedding strategy for arbitrary expert models that projects the models into an Euclidean space and preserves model similarities for subsequent clustering steps; we compare Prediction Loss to alternative embedding strategies and show its effectiveness for computing model distances in comparison to the existing Cross Performance distance measure.
- *ModelForge:* We introduce ModelForge, an expert model consolidation framework that composes three steps: model embedding, model clustering and model consolidation. With Prediction Loss, Euclidean distance and K-Means [8] clustering, it achieves an average accuracy improvement of $\sim 3.5\%$ compared to existing approaches; because the embedding provides transformations into Euclidean spaces, the framework is significantly faster and supports various clustering algorithms and performance optimizations.
- *Model Consolidation Datasets:* In addition to the energy consumption dataset, we create three further, diverse benchmarking datasets for model consolidation evaluations in the domains anomaly detection, weather forecasting, and price prediction; this enables consistent evaluations of consolidation approaches across different applications.

In Section II, we provide an overview of the background and related work in model clustering and consolidation; we specifically discuss the Cross Performance distance function and its weaknesses. Section III then proposes our main contribution *ModelForge* and the *Prediction Loss* embedding strategy for machine learning models in Euclidean space for clustering and consolidation. Section IV introduces the three additional evaluation datasets and discusses their background and construction. In Section V, we use these datasets to systematically evaluate the different embedding strategies. Finally, we summarize our findings in Section VI.

II. Background & Related Work

This section discusses hidden parameters and the challenges they pose in machine learning (Section II-A). Subsequently, we give an overview of metric spaces and their implications to clustering (Section II-B). Lastly, we discuss the existing *Cross Performance* distance measure and its properties (Section II-C).

A. Hidden Parameters

We target use-cases where hidden features that describe constant, setup-specific, but non-explicit information are prevalent. They cannot be learned directly, but influence the behavior of a trained model. Hidden features are constants in a specific training setup, i.e., for a single machine learning model, but may vary across different models. Because they are constant factors, their static impact on model outcomes is learned implicitly but it does not (necessarily) translate into other setups.

For example, we can use various sensors and settings, i.e. outside temperature, heating curve settings, and temperature target values to train energy consumption forecasting models for individual residential heating appliances. The actual forecasts are, then, also impacted by *hidden features*, such as building size and insulation, roof type, window coverage and glazing, inhabitant number etc. that are specific to individual heating systems, implicitly considered during model training, and usually not known by the provider of the energy consumption forecasting models. Because of these hidden features, consolidating all expert models in one global model greatly impacts the forecasts' accuracy; such a global model could, for instance, not distinguish an office building and a residential building. However, many installations share very similar properties and, therefore, hidden parameter values, such as the typical four-person family in an urban, modernized one-family home. It is, therefore, important to consolidate models carefully w.r.t. their hidden features.

B. Metric Spaces

A metric space provides a mathematical framework for measuring distances between individual elements of a set.

Definition 1: Metric Spaces: Let X be a set of objects with a distance function $d : X \times X \to \mathbb{R}^+$. d is a metric iff it satisfies the following properties for all $x, y, z \in X$:

$$\begin{aligned} d(x,y) &\geq 0 && \text{Non-Negativity} \\ d(x,x) &= 0 && \text{Identity} \\ d(x,y) &= d(y,x) && \text{Symmetry} \\ d(x,z) &\leq d(x,y) + d(y,z) && \text{Triangle-Inequality} \end{aligned}$$

While various distance functions may be formulated to quantify dissimilarity between data points, only those satisfying all four properties qualify as proper metrics.

Different clustering algorithms, including hierarchical clustering [9] and spectral clustering [10], are capable of operating without distance metrics. In many specialized fields, such as time series clustering, non-metric distances, e.g., Dynamic Time Warping [11] or shape-based distance measures [12], are commonly utilized. Nonetheless, proper distance metrics possess mathematical properties that offer substantial computational benefits:

Reliable convergence: The convergence guarantees of numerous clustering algorithms depend strongly on metric properties. For instance, K-means [8] leverages Euclidean distance, a proper metric, that satisfies the triangle inequality to reliably converge to a local optimum [13]. Employing non-metric distance functions may result in oscillatory behavior where the algorithm fails to converge [14].

Spatial consistency: Metric properties ensure spatial consistency within the feature space. The triangle inequal-

ity, in particular, formalizes the intuitive notion that if point x is close to point y, and y is close to point z, then x and z cannot be arbitrarily distant. This property is essential for the formation of coherent clusters that align with human intuition regarding grouping.

Computational optimizability: Many algorithmic optimizations in clustering leverage metric properties through the use of index structures like ball-trees [15] and KD-trees [16], which both rely fundamentally on the triangle inequality to prune the search space. Clustering algorithms like DBSCAN [17] utilize these indices to reduce the number of distance calculations required during neighborhood queries to reduce runtime.

C. Cross Performance

Cross performance [1] is a distance measure for arbitrary machine learning models that defines the distance of two models to be the average loss of the two models on each other's test sets. Formally, the distance is defined as follows:

Definition 2: Cross Performance: For any two models m_i and m_j with their respective test sets (X_i, Y_i) and (X_j, Y_j) and a loss function l, the *Cross Performance* $d(m_i, m_j)$ of m_i and m_j is calculated as:

$$d(m_i, m_j) = \frac{1}{2}(l(m_i(X_j), Y_j) + l(m_j(X_i), Y_i)). \quad (1)$$

To analyze the properties of $d(m_i, m_j)$, we now review Cross Performance in terms of metricity according to Definition 1.

a) Non-Negativity: If l is non-negative, e.g., the mean squared error (MSE), then $d(m_i, m_j) \geq 0$ and d fulfills non-negativity.

b) Identity: The identity property requires $d(m_i, m_i) = 0$, which holds only if $l(m_i(X_i), Y_i) = 0$. This requires a perfect prediction model, which, in practice, is rarely the case. Hence, d violates identity.

c) Symmetry: With simple rearrangements of the definition of d, it can easily be shown that $d(m_i, m_j) = d(m_j, m_i)$. Therefore, d fulfills symmetry.

d) Triangle Inequality: Because m_i and m_j as well as (X_i, Y_i) and (X_j, Y_j) are arbitrary, we can choose $m_1(x) = 2x$, $m_2(x) = x$, and $m_3(x) = -2x$ with $(X_1, Y_1) = (\{5\}, \{10\})$, $(X_2, Y_2) = (\{0\}, \{0\})$, and $(X_3, Y_3) = (\{5\}, \{-10\})$. If we, then, choose the MSE for l, we get $d(m_1, m_2) = 12.5$, $d(m_2, m_3) = 112.5$, and $d(m_1, m_3) = 400$. Since $400 > 12.5 + 112.5$, d violates the triangle inequality.

Because d violates the identity and triangle inequality properties, Cross Performance is not a metric. For this reason, Cross Performance works only with clustering algorithms, such as hierarchical clustering, that do not require metrics and are usually more expensive to calculate. The distance matrix calculation for hierarchical clustering, specifically, has a quadratic runtime in $\Theta(\frac{n^2-n}{2})$ with n objects and, hence, many expensive Cross Performance distance calculations that make the approach inapplicable to large datasets. The metric violations also impact the accuracy of Cross Performance, allowing for a more accurate, metric-based distance calculation in *ModelForge*.

We also differentiate model consolidation from model compression techniques, such as knowledge distillation or pruning. While sharing the goal of efficiency, these methods typically apply to single, over-parameterized models (often neural networks), whereas ModelForge is model-agnostic and designed to consolidate a large portfolio of heterogeneous expert models, addressing a distinct M-to-N consolidation problem.

III. ModelForge

In this section, we present our machine learning model consolidation approach *ModelForge*. *ModelForge* takes a set of expert machine learning models M as input and groups similar models with the aim of consolidating every group into one representative model. All models $m_i \in M$ have the same shape of input feature(s) x and target variable(s) y, i.e., $m_i : x \rightarrow y$. For simplicity and w.l.o.g. we consider $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$ in this section, but other feature- and prediction-domains, such as categorical features and probabilistic targets, are also supported in *ModelForge*.

We first provide an architectural overview of the consolidation framework (Section III-A). Because we aim to create a clustering approach based on metric spaces, we focus on an embedding-based distance calculations in this framework. To find an effective embedding strategy, we introduce three possible embedding strategies for machine learning models including *Prediction Loss*, which is the strategy that will ultimately perform best (Section III-B). All three strategies rely on the identification of either representative reference models (Set-based) or representative data points (Point-based). For this reason, we subsequently introduce possible selection strategies including the eventually best performing set-based *Uniform Target Entropy* strategy (Section III-C). Although *ModelForge* supports any combination of embedding and selection strategies, we show later in Section V that Prediction Loss embedding with Uniform Target Entropy selection is the overall most effective model consolidation strategy.

A. Architectural Overview

The *ModelForge* framework consists of three stages, which are Embedding, Clustering and Consolidation as depicted in Fig. 1. Given a dataset of machine learning models and their associated training and testing data, *ModelForge* first leverages a similarity preserving embedding strategy to extract a vector representation for each model that effectively projects the model into Euclidean space; models with similar behavior, i.e., similar hidden features are placed nearby. Then, *ModelForge* applies a clustering algorithm, which is K-Means with Eucledian distance in our setup, on the embedded models to group similar models (and their associated training and testing data). *ModelForge* finally consolidates each identified

cluster of similar expert models into one cluster model by merging the training data of the contained models and, then, simply training a new model with the same architecture as the expert models but on the combined training data. This cluster model serves as representative for all setups in this cluster. Slight differences in the hidden features can make this cluster model a bit weaker than the individual expert models, but as we will later see in Section V the increased amount of training data usually compensates for this disadvantage such that the increment in loss is small.

B. Embedding strategies

Fig. 2 taxonomizes consolidation approaches, showing how ModelForge focuses on creating output-based embeddings of models rather than using pairwise distances, such as Cross Performance. These approaches can generally be classified into input-, structure-, and output-based categories, on the condition whether they ascertain model similarity via the input data of a model, its internal architecture, or its resulting output. Prior research [1] has already explored the former two methodologies, revealing their inadequacy in producing accurate model consolidations. This shortcoming prompted the introduction of Cross Performance, an output-based pairwise distance measure. *ModelForge*, therefore, also builds upon output based model clustering approaches. To create metric spaces for the clustering and, in this way, improve the clustering performance, it specifically focuses on the family of strategies that produces model *embeddings*. We introduce three strategies for embedding a model based on its outputs: leveraging its raw prediction (*Prediction Value*, i.e., what it predicts), its prediction error (*Prediction Loss*, i.e., how wrong it is), or its feature attributions (*Prediction Explanation*, i.e., why it made the prediction). Each of these can be combined with either a point-based selection strategy or a set-based selection strategy, which we later discuss in more detail in Section III-C.

For the *point-based selection* strategy, we select (or sample) from all training sets of all models a set of s training data points $(x^{(j)}, y^{(j)})$, where $x^{(j)} \in \mathbb{R}^{|N|}$ is an $|N|$ dimensional feature vector and $y^{(j)} \in \mathbb{R}$ the corresponding ground truth target value. With $X = [x^{(1)}, x^{(2)}, ..., x^{(k)}]$ we describe the k selected feature vectors and with $Y = [y^{(1)}, y^{(2)}, ..., y^{(k)}]$ the corresponding ground truth values.

For the *set-based selection* strategy, we select from all models a set of s different models m_j with their entire respective training sets $(X^{(j)}, Y^{(j)})$. Each $X^{(j)} = [x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(n)}]$ is the set of feature vectors of model m_j and $Y^{(j)} = [y_j^{(1)}, y_j^{(2)}, \ldots, y_j^{(n)}]$ the corresponding ground truth values. The set of the s selected feature vector sets is denoted as $\hat{X} = [X^{(1)}, X^{(2)}, \ldots, X^{(s)}]$ and the set of the s corresponding ground truth training values is denoted as $\hat{Y} = [Y^{(1)}, Y^{(2)}, \ldots, Y^{(s)}]$.

1) Prediction Value: The concept of employing predictions for embedding a machine learning model is based on a fundamental principle of supervised learning: A model's outputs encapsulate the learned mapping from the input features to the target variable. These predictions represent the essence of the learned behavior of a model, thus providing a functional signature of the model's performance.

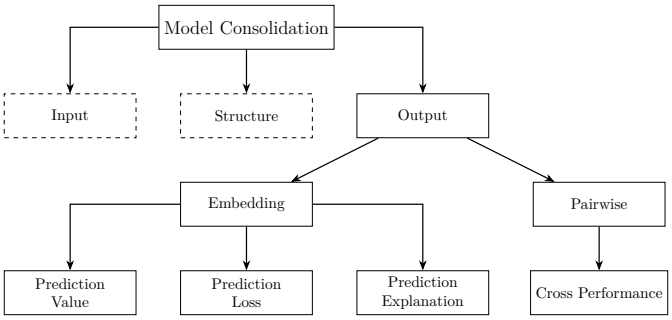

Fig. 2. Taxonomy of model consolidation approaches.

When two models produce similar predictions across a shared set of inputs, we can reasonably infer that these models have learned similar mappings of the feature values to the target variable, regardless of their architectural differences.

The *Prediction Value* embedding e_i of any model m_i is constructed for a point-based selection strategy by calculating its predictions over the set of selected points $X = [x^{(1)}, x^{(2)}, ..., x^{(s)}]$. Hence, the embedding is defined as:

$$e_i = [e_i^{(1)}, e_i^{(2)}, ..., e_i^{(s)}] \text{ with } e_i^{(j)} = m_i(x^{(j)}) \tag{2}$$

where $e_i \in \mathbb{R}^s$ is the resulting embedding vector for model m_i. Each component of this embedding corresponds to the model's prediction on a specific point from our selected set.

For a set-based selection strategy, s different models with their associated training sets are selected among all models. With $\hat{X} = [X^{(1)}, X^{(2)}, \ldots, X^{(s)}]$, the embedding is computed as the mean prediction value of m_i on each $X^{(j)}$:

$$e_i = [e_i^{(1)}, e_i^{(2)}, ..., e_i^{(s)}] \text{ with } e_i^{(j)} = \frac{1}{|X^{(j)}|} \sum_{a=1}^{|X^{(j)}|} m_i(x_j^{(a)}) \tag{3}$$

2) Prediction Loss: The Prediction Value embedding captures a model's behavior through its raw outputs; hence, it does not account for how well these predictions align with the ground truth. The *Prediction Loss* embedding strategy addresses this limitation by incorporating performance metrics into the model's representation space.

The Prediction Loss strategy is predicated on the principle that models should be differentiated not only by what they predict but also by how well they predict. By embedding models in a space defined by their errors rather than their raw predictions, this approach creates a performance-aware representation that naturally separates high-performing models from lower-performing ones across various input scenarios.

Let $l : \mathbb{R}^s \times \mathbb{R}^s \to \mathbb{R}$ be a loss function that quantifies the discrepancy between a model's prediction $m_i(x^{(j)})$ and the actual observed value $y^{(j)} \in \mathbb{R}$. In conjunction with the point-based selection strategy, the Prediction Loss embedding for a model m_i is, then, formally defined as:

$$e_i = [e_i^{(1)}, e_i^{(2)}, ..., e_i^{(s)}] \text{ with } e_i^{(j)} = l([m_i(x^{(j)})], [y^{(j)}]) \quad (4)$$

where $e_i \in \mathbb{R}^s$ is the resulting embedding vector for model m_i, with each component corresponding to the model's prediction error on a specific sample.

With the set-based selection strategy and the selected multi-sets $\hat{X}$ and $\hat{Y}$, the Prediction Loss is computed by calculating a model m_i's loss on every selected training set $(X^{(j)}, Y^{(j)})$ with the loss function l:

$$e_i = [e_i^{(1)}, e_i^{(2)}, ..., e_i^{(s)}] \text{ with } e_i^{(j)} = l([m_i(x_j^{(1)}), \ldots, m_i(x_j^{(n)})], Y^{(j)}) \quad (5)$$

3) Prediction Explanation: The Prediction Explanation strategy leverges Shapley values [18]–[20], from cooperative game theory, providing an approach to feature attribution in machine learning models. They quantify the contribution of each individual feature to a model's prediction for a specific data point.

In the context of model embeddings, Shapley values serve a dual purpose: They not only explain feature importance for a given input but also provide a mechanism to embed models themselves. The key insight is that models with similar behavior should attribute similar importance to features for the same input, resulting in embeddings where similar models m_i and m_j have only small distances between them when evaluated on the same data.

For a model m_i, the Shapley value $\phi_{i,a}$ for feature a when explaining a prediction is generally defined as [20]:

$$\phi_{i,a} = \sum_{S \subseteq N \setminus \{a\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [m_i(S \cup \{a\}) - m_i(S)] \quad (6)$$

where N is the complete set of all features, S represents a subset of features excluding feature a and $m_i(S)$ denotes the model's effective output when considering only features in set S. The combinatorial weights $\frac{|S|!(|N|-|S|-1)!}{|N|!}$ ensure fair attribution across all possible feature combinations [20].

A critical aspect of computing Shapley values is determining the model's effective output $m(S)$ when features are excluded from a combination S. These tests are realized by using a background distribution. Let $X = [x^{(1)}, x^{(2)}, ..., x^{(s)}]$ be a selected set of s points with each $x^{(j)} \in \mathbb{R}^{|N|}$ for which we wish to compute Shapley values. Moreover, let $\tilde{X} = [\tilde{x}^{(1)}, \tilde{x}^{(2)}, ..., \tilde{x}^{(n)}]$ be the background distribution consisting of n reference points. For any subset $S \subseteq N$, we define a mixed instance z_S based on $x^{(j)}$ and $\tilde{X}$. The a-th component of z_S, denoted $z_S^{(a)}$, is defined as:

$$z_S^{(a)} = \begin{cases} x_a^{(j)} & \text{if } a \in S \\ \tilde{x}_a^{(r)} & \text{if } a \notin S \end{cases} \quad (7)$$

where $x_a^{(j)}$ is the value of feature a in instance j, $\tilde{x}_a^{(r)}$ is the value of feature a in a randomly chosen background sample $\tilde{x}^{(r)}$.

Using the expected value approach, the model evaluation $m_i(S)$ is computed as the expected value over the background distribution [20]:

$$m_i(S) = \mathbb{E}_{\tilde{x} \sim \tilde{X}}[m_i(z_S)]. \quad (8)$$

Note that because z_S depends on $x^{(j)}$, the resulting $m_i(S)$ and therefore the calculated Shapley values $\phi_{i,a}$ are specific to the instance $x^{(j)}$, which will be further denoted as $\phi_{i,a}^{(j)}$.

For a point-based selection strategy, we now compute our embedding vector $e_i \in \mathbb{R}^s$ for a model m_i, derived from the analysis of the data points $x^{(j)} \in X$, as the mean of the features' Shapley values to gain an understanding of the overall sample importance for each instance $x^{(j)}$:

$$e_i = [e_i^{(1)}, e_i^{(2)}, ..., e_i^{(s)}] \text{ with } e_i^{(j)} = \frac{1}{|N|} \sum_{a=1}^{|N|} \phi_{i,a}^{(j)} \quad (9)$$

For a set-based selection strategy with $\hat{X} = [X^{(1)}, X^{(2)}, \ldots, X^{(s)}]$, the Prediction Explanation embedding for model m_j is, then, computed as mean of the feature importances of each sample within $X^{(j)}$:

$$e_i = [e_i^{(1)}, e_i^{(2)}, ..., e_i^{(s)}] \text{ with } e_i^{(j)} = \frac{1}{|X^{(j)}| \cdot |N|} \sum_{l=1}^{|X^{(j)}|} \sum_{a=1}^{|N|} \phi_{i,a}^{(l)} \quad (10)$$

C. Selection strategies

This section outlines concrete point- and set-based selection strategies for the embedding strategies described in Section III-B. These selection strategies are used in *ModelForge* to choose training points X and Y or training sets $\hat{X}$ and $\hat{Y}$, respectively. All strategies except the *random* strategies use the ground truth target values (potential Y and $\hat{Y}$ values) in their selection heuristics.

1) Point-Based: Point-based selection strategies identify a set of individual data points from the full corpus of training data of all models. In the following, we present five point-based selection strategies.

a) Random: The random selection strategy chooses n individual data points randomly. This approach provides an unbiased representation of the overall data distribution.

b) Min/Max: This strategy selects s samples based on the extremes of the target variable distribution. More specifically, the Min/Max strategy selects the s samples with the lowest/highest target values. This approach helps characterize model behavior at the boundaries of the prediction space, where models often exhibit the greatest variability in performance.

c) Uniform: The uniform selection strategy first sorts all data points according to their target values, then selects s samples at regular intervals across this sorted distribution. This ensures a representative coverage across the entire range of target values, providing insight into the behavior of the model across the complete spectrum of prediction scenarios.

d) Percentile Random: This strategy implements a two-step selection process: First, it filters the combined training data to include only points falling between the 25th and 75th percentiles of the target variable distribution. Then, it randomly selects s data points from this filtered set. By focusing on the interquartile range, this approach emphasizes model behavior in the most common cases while reducing the influence of points at the boundaries of the prediction space.

2) Set-Based: Set-based selection strategies sample entire training sets from the corpus of the training sets of all models. In the following, we discuss ten strategies for this selection objective:

a) Random: The Random selection strategy randomly samples s different training sets from the corpus of models. This approach provides a method for creating diverse sample sets without introducing specific biases.

b) Min/Max/Uniform Median/Variance/Entropy: The Min/Max/Uniform Target strategies first aggregate the training sets by their target values y. The aggregations can use either *median*, *variance*, or *entropy*. Afterwards, the strategies sort the training sets by these aggregates to, then, select the s sets with the lowest aggregates (Min), the highest aggregates (Max), or a uniform distribution of aggregates (Uniform).

IV. Crafting Model Datasets

In the following, we describe the construction of four machine learning model datasets, which we use in Section V for validation: The dataset *Heating* contains energy consumption models, *Anomaly* contains timeseries anomaly detection models, *Weather* contains weather station postprocessing models and *Housing* contains house pice prediction models. Table I provides an overview of the datasets with their metadata and properties. In this section, we discuss the datasets and their models in more detail; further details are listed in our Github repository.[1].

A. Heating Dataset

The Heating dataset consists of timeseries data of 919 heating devices, including oil-/gas boilers as well as heat pumps. The timeseries contain the daily aggregated measured energy consumption that was used for residential heating along with additional sensor recordings and features. The features include i.a. outdoor temperature, mean target supply temperature, and yearly seasonality. All features have been z-normalized and, then, used to train 919 Random Forest [25] and Gradient Boosting Tree [26] models to predict the daily energy consumption of the different heating devices.

B. Anomaly Dataset

The Anomaly dataset comprises univariate timeseries collected from the NASA-MSL [2], NASA-SMAP [2], IOPS [3], and KDD-TSDA [4], [5] datasets. We removed timeseries that had no anomaly in their training- or test set, which results in a total number of 357 timeseries. Each datapoint within these timeseries is labeled with a binary indicator: 1 for anomalous and 0 for normal. With the time series, we trained 357 XGBoost [27] models with a window size of 50 observations in forecasting anomaly scores. For a threshold agnostic evaluation, we measure the detection quality with the ROC-AUC metric [21], [22].

C. Weather Dataset

The Weather dataset serves to enhance temperature forecasts at a height of 2 meters. It contains 48-hour lead time predictions derived from the European Center for Medium-Range Weather Forecasts (ECMWF) 50-member ensemble, as presented in [6]. The predictions are sourced from the THORPEX Interactive Grand Global Ensemble (TIGGE) archive [28] and are complemented by observed weather station data provided by the Deutscher Wetterdienst (DWD). We used 537 surface weather stations across Germany and all their meteorologically relevant predictor variables with 2 meter temperature observations provided by the DWD to train 499 models in postprocessing and improving ECMWF forecasts. More specifically, we trained probabilistic XGBoost models with Natural Gradient Boosting [29] in predicting the mean and standard deviation of a gaussian target distribution in the years 2007 to 2015; the training excluded 38 weather stations with missing observational data and the recordings of 2016 (182,218 samples) are used for validation. The performance of the models is measured with CRPS [23], [24]. In contrast to the datasets original publication [6], we hide any station specific information (e.g. geolocation or height) during the training process to deliberately introduce hidden features in the setup.

D. Housing Dataset

The Housing dataset originates from a dataset published by the real estate internet platform Immoscout24 and the German Forschungsdatenzentrum Ruhr (FDZ Ruhr). It includes all residential properties published on Immoscout24 between January 2007 and June 2024. In total it contains 17,528,584 houses with various characteristics, such as price, size and broad location on a 1 km^2 grid. We further enriched the data with additional information made available by the German Federal Statistical Office and other official authorities; these include information about the labor market [30], house price indices [31], construction industry [32], wage index [33], economy data [34], building land prices [35], building permits [36], consumption propensity [34], financial interest rates [37], age structure [38] and population development [39]. Categorical features with more than two unique values, such as property type, energy efficiency class or equipment standard, are encoded using dummy variables. We finally partitioned the data by county and, then, trained 389 XGBoost [27] models (one per county) to predict house prices. By hiding

[1] https://github.com/UMR-Big-Data-Analytics/ModelForge

TABLE I
DATASETS FOR EVALUATION WITH THEIR ASSOCIATED METADATA.

Name	Source	Applied Loss	No. models	Training Data			
				No. Features	Min	Mean	Max
Heating	Proprietary	MAE	919	6	58	293	937
Anomaly	[2]–[5]	ROC AUC [21], [22]	357	50	1,435	85,604	1,149,900
Weather	[6]	CRPS [23], [24]	499	34	515	3,522	3,651
Housing	[7]	MAPE	389	109	1,197	32,757	6,371,308

all geographic details of individual real estates, such as cities, counties or geo-locations, all trained models are effectively based on hidden features.

V. Experiments

Our experiments demonstrate that *ModelForge* constructs meaningful clusters for the consolidation of machine learning models. First, we describe the experimental setup (Section V-A). Then, we assess the effectiveness of the different embedding strategies with varying parameterizations (Section V-B) and examine the effect of the embedding size (Section V-C). Afterwards, we analyze the space associated with our suggested *Prediction Loss* embedding, which shows that the space is benign and, as a result, not susceptible to the curse of dimensionality [40] (Section V-D).

A. Experimental Setup

To evaluate our approach, we use the measure from [1] to compare the consolidated model's prediction error against the average error of its constituent expert models, assessing the performance trade-off. This comparison allows us to assess how the consolidated models perform in contrast to the original expert models. Because the proposed evaluation measure used the mean absolute error (MAE) as loss function l, we adapt it to use the respective loss function of the individual expert models: For each cluster C_i and a given loss function l, we compute the error μ_i of the consolidated cluster model m'_i across all test sets (X_j, Y_j) of the expert models M within cluster C_i:

$$\mu_i = \frac{1}{|C_i|} \sum_{m_j \in C_i} l([m'_i(x_j^{(1)}), \ldots, m'_i(x_j^{(n)})], Y_j) \tag{11}$$

Because l is a loss function, we compute the inverse of l denoted as l^{-1}, if for some specific l larger values indicate better scores, e.g. for ROC AUC [21], [22].

To derive an aggregate performance measure for a clustering, we compute the weighted mean over all clusters. The weight corresponds to the cluster size, with $N = \sum_{i=1}^{k} |C_i|$ being the total number of models, k denoting the number of clusters, and $|C_i|$ being the size of cluster i:

$$\mu = \frac{1}{N} \sum_{i=1}^{k} |C_i| \cdot \mu_i \tag{12}$$

Because the magnitudes of loss functions l vary for different datasets, we divide the consolidation score μ for a clustering with k clusters with the score of a baseline clustering that also used k clusters. We use the geometric mean to aggregate these relative scores for all chosen k, because for relative scores, the arithmetic mean can lead to deceptive conclusions [41].

We chose *Cross Performance* as baseline method in our evaluation. Because Cross Performance produces a distance matrix, it uses *hierarchical clustering* with complete linkage [1]. Our embedding strategies, in contrast, use *K-Means* [8] for clustering. We note that computing the full distance matrix for the Heating dataset (n=919) took over 207 minutes on our test machine, whereas generating the ModelForge embeddings for the same set took less than 3 minutes. Further performance tests are skipped here due to space limitations and can be viewed on our Github repository. For all experiments, we consolidate each cluster into one model by merging the training sets of all its models and, then, training a new model.

B. Effectiveness of Embedding Strategies

We begin by showing that *Prediction Loss* indeed captures the behavior of models in its embeddings better than the other embedding strategies. For this experiment, we combine every embedding strategy with every selection strategy. For each datasets, i.e., *Anomaly*, *Heating*, *Housing*, and *Weather*, we run every embedding-selection combination 50 times with varying numbers of target clusters k. We choose k evenly spaced in the interval of 0.5% to 25% of the respective datasets' size. For percentages higher than 25%, clusters become too small to observe meaningful differences between the embedding strategies; for percentages below 0.5%, the clusters usually cannot capture the hidden features any more as they become too large. As the performance is not sensitive to the embedding size s (see Section V-C), we use a constant $s = 10$ for the experiments. Because for every dataset the magnitude of the evaluation score μ differs due to different loss functions (ROC AUC^{-1}, MAE, MAPE, CRPS), this experiment considers the relative score improvement/deterioration w.r.t. the score of the already existing *Cross Performance* strategy. We show the relative scores for the embedding-selection combinations aggregated over all four datasets in Fig. 3 and per dataset in Fig. 4. The embedding strategies are color-coded and the selection strategies are structure-coded. The $\star$-symbol marks the best embedding-selection combination per embedding strategy and scores that exceed the inspection range of $\pm 5\%$ relative score are annotated to the right of the bars.

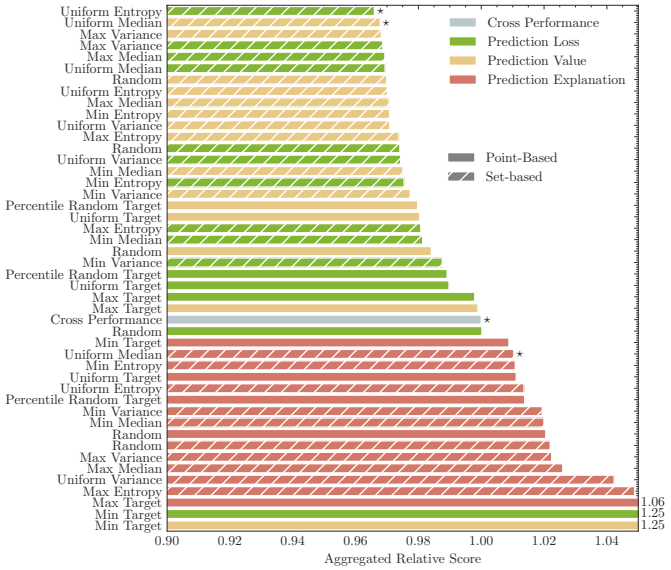

Fig. 3. Relative μ scores of all embedding-selection strategies aggregated over all four datasets. Lower values indicate better performance.

Fig. 3 shows that both *Prediction Value* and *Prediction Loss* embeddings perform in general better than *Cross Performance*. We attribute this improvement to the fact that distance calculations on these embeddings have metric properties and in particular satisfy the triangle inequality, which *Cross Performance* does not fulfill. The *Prediction Explanation* embedding shows clearly worse scores than all other strategies. The explainability scores consequently do not characterize the properties of the models as well as the prediction values or losses. The performance breakdowns per dataset in Fig. 4 show the same general insights. For *Anomaly* and *Weather*, however, the choice of a proper selection strategy is important to improve upon the performance of *Cross Performance*: First of all, the measurements show that *set-based* selections are, in general, superior to *point-based* selections, which is because these embeddings consider more samples and taking coherently entire training sets is also often advantageous. Furthermore, *Max* selections perform better than *Min* selections, because maxima help the embeddings to capture model behaviors better in generally more difficult settings and minima often tend towards trivial predictions, such as time series with hardly any anomalies or energy consumption predictions in summers; overall, however, *Uniform* sampling delivers the most characteristic embeddings, as both *Max* and *Min* introduce a bias towards overly noisy or silent training sets. The aggregation, which is either *Entropy*, *Variance*, or *Median*, does not play a significant role when being used in conjunction with *Max* or *Uniform* selection; all three seem to characterize the target value distributions similarly well. Across all measurements, *Prediction Loss* with *set-based Uniform Entropy* selection is the on average best performing and most stable clustering strategy.

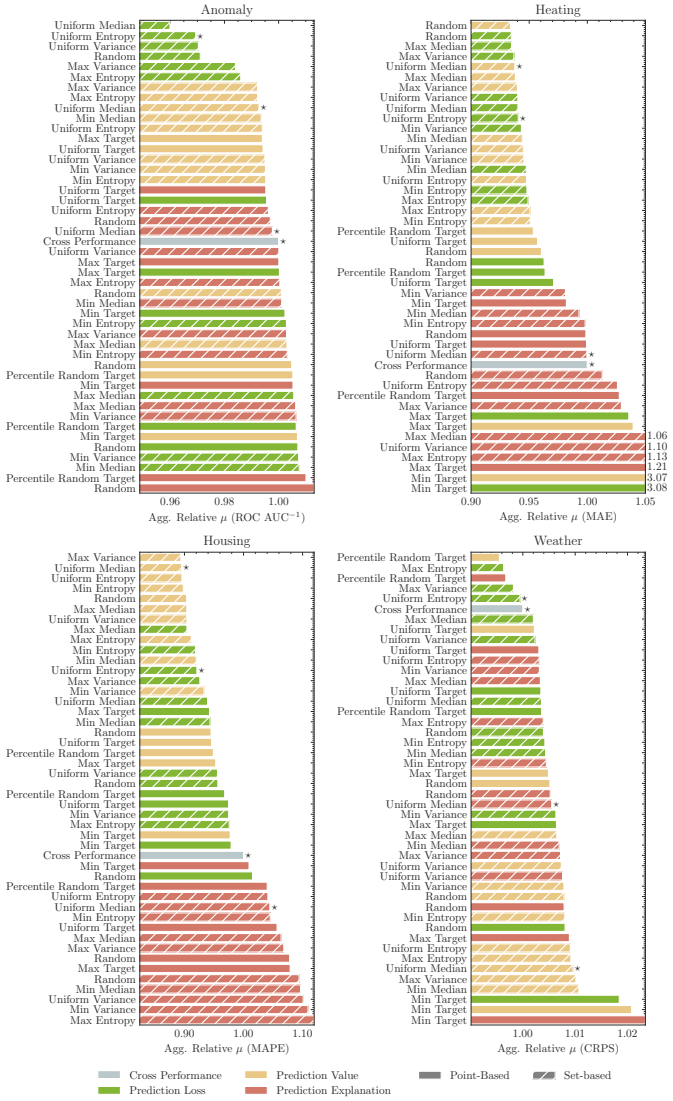

Fig. 4. Relative μ scores of all embedding-selection strategies for each dataset. Lower values indicate better performance.

To evaluate the impact of the model consolidation on the effectiveness of the models, we now measure the absolute μ scores for every embedding strategy (with their respective best selection strategies) on all four datasets for increasing numbers of clusters k. The results are shown in Fig. 5. For solid lines, we re-trained new models for every cluster; for the dashed line of the *Prediction Loss* strategy, we selected the most effective model per cluster as a representative. These dashed lines demonstrate that re-training produces clearly more effective models than model selection. Overall, the measurements show that the μ scores tend to improve (= decrease) with an increasing number of clusters, i.e., models. Due to many hidden features especially in the *Anomaly* and *Housing* datatsets, more specialized expert models yield better performances. For *Heating* models, we can reduce the number of models

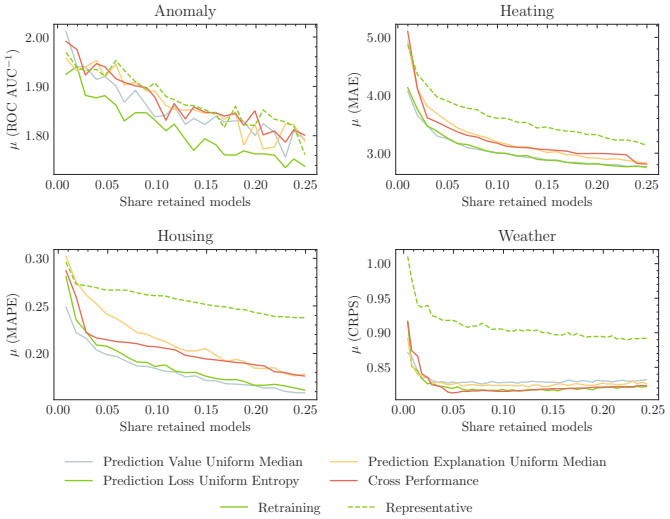

Fig. 5. Absolute μ scores of the three embedding strategies and *Cross Performance* with their respective best selection strategies for increasing numbers of clusters k in 0.5% to 25% of the dataset sizes.

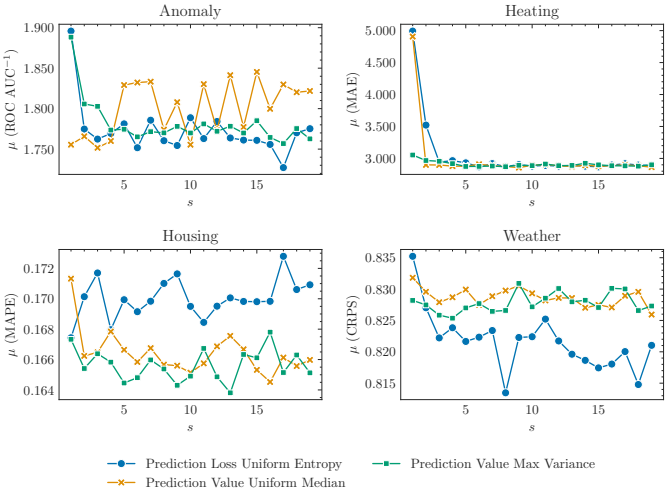

Fig. 6. Absolute μ scores of the three best embedding and selection strategy combinations with a fixed number of clusters for different embedding size s.

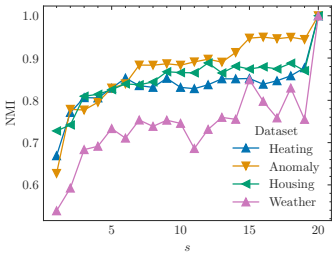

Fig. 7. Normalized Mutual Information (NMI) of clusterings across four datasets for different embedding size s w.r.t. a base clustering with $s = 20$.

to 10% compared to 25% without really loosing accuracy and the *Weather* models even improve their accuracy when reduced to $5-10\%$, because the cluster models can be trained on larger training sets than the expert models. So in general, model consolidation might sacrifice a little accuracy, but for a significant reductions in the numbers of models; the appropriate choice of k is use case specific, though. Regardless of k, *Prediction Loss* is consistently performing best (or almost best).

C. Influence of Embedding Size

We will now investigate the influence of the embedding size s on the performance scores μ to demonstrate the insensitivity of the embedding strategies towards s. For this, we again consider every embedding strategy with its respective best selection strategy, calculate the absolute μ scores on all four datasets with varying embedding sizes s, and plot the results in Fig. 6. Based on Fig. 5, we set k to represent the share of 0.2 of the *Anomaly* models, 0.15 of the *Heating* models, 0.2 of the *Housing* models, and 0.05 the of *Weather* models.

The measurements in Fig. 6 first of all show that the general noise of the consolidation is higher than the impact of the embedding length s. Only for very small embeddings $s \leq 3$, we observe some significant performance drops indicating that aggressively small embeddings can be detrimental to performance. For increasing embedding sizes s, we can interpret slightly positive or negative trends into the noisy measurements of the different embedding strategies, but with $s > 3$ the concrete size is ultimately not that important: For the four consolidation scenarios, the noise-removed improvement / deterioration trends affect the μ scores only slightly with increasing s. Hence, *ModelForge* can achieve meaningful clusterings with relatively small embedding sizes and, hence, comparatively low computation and memory costs.

To further understand the stability of the consolidation scores, we turn to similarity of the resulting clustering when varying s. For this, we extracted the cluster labels of the models computed with *Prediction Loss* and *Uniform Entropy* as selection strategy for varying s and computed the Normalized Mutual Information (NMI) between the clustering and a baseline clustering with $s = 20$. Higher NMI values indicate greater clustering similarity. The results in Fig. 7 indicate that the similarity with the base clustering rises sharply until $s = 5$ and then settles at a high level or gaining minor similarity for all datasets and, hence, explaining the observation that s has only minor effects on the consolidation scores because the produced clusterings are similar to each other.

D. Inspection of the embedding spaces

Our final experiment analyses the embedding space of the *Prediction Loss* strategy showing that it is well-natured and not affected by the curse of dimensionality [40]. The curse of dimensionality refers to various phenomena that can arise when analyzing data in high-

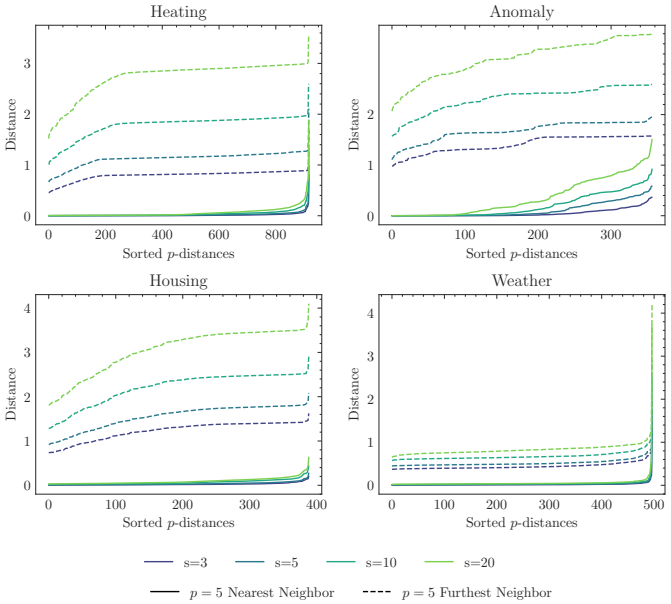

Fig. 8. $NN(p)$-distance graphs ($p = 5$) of the *Prediction Loss* embedding with *Uniform Entropy* selection at varying embedding sizes $s \in \{3, 5, 10, 20\}$ on four datasets. Solid lines are distances to nearest neighbors and dashed lines are distances to furthest neighbors.

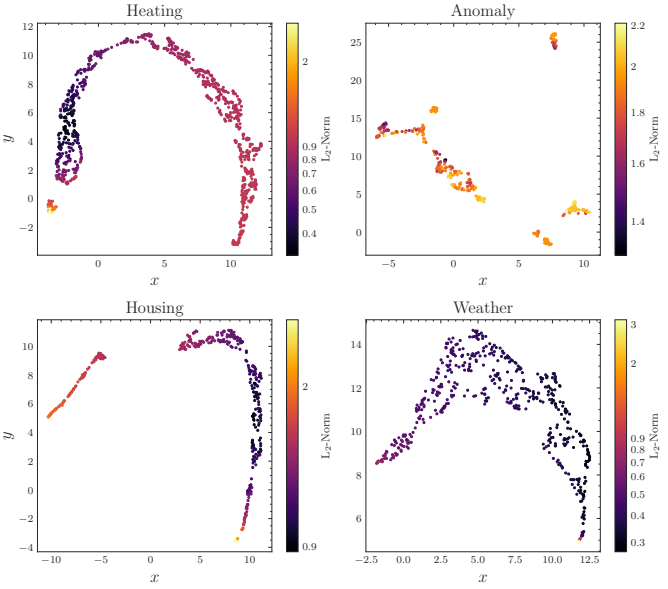

Fig. 9. 2D UMAP visualizations of the embedding spaces of *Prediction Loss* embedding with *Uniform Entropy* selection ($s = 10$).

dimensional spaces. The volume of the space grows exponentially with the number of dimensions, causing data to become sparse and distances between points to potentially become meaningless due to the loss of numerical contrast; hence, distance-based algorithms may degenerate. This is also known as concentration effect [42].

Fig. 8 shows the $NN(p)$-distance graphs across four datasets at varying embedding sizes $s \in \{3, 5, 10, 20\}$ using *Uniform Entropy* selection. $NN(p)$ is the distance to the p-th nearest neighbor, with $p = 5$ in our analysis. Solid lines represent distances to the p-th nearest neighbors, while dashed lines show distances to k-th furthest neighbors. All embeddings are scaled to fit within a unit hypercube of side length 1. The x-axis indicates the sorted distances of each point to its p-th nearest/furthest neighbor. We chose $p = 5$ because it provides a good balance between local detail and global structure. For all datasets and values of s, a notable difference between the p-th furthest and nearest neighbor can be observed, which indicates that distances are still meaningful because for any given query point we can clearly differentiate near and far away other points.

Fig. 9 shows 2D UMAP [43] projections of the *Prediction Loss* embedding spaces with *Uniform Entropy* and $s = 10$ for all four datasets on a 2D plane. The coloring indicates the L_2-norm of the embeddings in the original space; the log-scale increases the contrast of the coloring. In the case of the *Prediction Loss* embedding, the L_2-norm can be interpreted as a proxy for the models' losses on the evaluated training sets during construction time; a higher L_2-norm corresponds to higher prediction errors on the used training sets. For the *Heating*, *Housing* and *Weather* dataset, the visualization reveals a U-shaped structure, whereas the *Anomaly* dataset exhibits several disconnected regions. For the former datasets, embeddings with larger L_2-norms and, hence, larger average losses can be found at both ends of the U, while better performing models with smaller L_2-norms are located around the bottom of the U. To understand this phenomenon, we took five models from both ends of the U and tested them on respective training sets that were used for constructing the embedding. The results showed that the two groups can be distinguished by the average difference of prediction and the actual values exhibiting negative values for one group and positive values for the other group. Consequently, models seem to diverge into two main behavioral paths: Some models systematically overestimate, while others underestimate the target variable. The observation shows that the embedding spaces do preserve implicit model properties.

VI. Conclusion

In this paper, we introduced *ModelForge*, a system for the embedding, clustering and consolidation of machine learning models. The embedding-based consolidation approach can utilize different embedding and selection strategies. Our evaluations, however, demonstrate that *Prediction Loss* embedding with *Uniform Entropy* selection achieves the most accurate consolidated models and also clearly outperforms related work.

Acknowledgment

We express gratitude to Viessmann IT Service GmbH's management for their support and permission to present this work and to FDZ Ruhr for supplying the Immoscout24 dataset.

References

[1] F. Siepe, P. Wenig, and T. Papenbrock, "A Few Models to Rule Them All: Aggregating Machine Learning Models," in *Lernen, Wissen, Daten, Analysen (LWDA)*, ser. CEUR Workshop Proceedings, vol. 3630, 2023, pp. 506–520.

[2] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, Jul. 2018, pp. 387–395.

[3] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-Series Anomaly Detection Service at Microsoft," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, Jul. 2019, pp. 3009–3017.

[4] R. Wu and E. J. Keogh, "Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress," Sep. 2020.

[5] E. Keogh, T. Dutta Roy, U. Naik, and A. Agrawal. (2021) Multi-dataset Time-Series Anomaly Detection Competition. [Online]. Available: https://compete.hexagon-ml.com/practice/competition/39/

[6] S. Rasp and S. Lerch, "Neural Networks for Postprocessing Ensemble Weather Forecasts," *Monthly Weather Review*, vol. 146, no. 11, pp. 3885–3900, 2018.

[7] RWI-Leibniz-Institut Für Wirtschaftsforschung and ImmobilienScout24, "RWI Real Estate Data - Houses for Sale - SUF," 2024.

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

[9] S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, Sep. 1967.

[10] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14. Cambridge, MA, USA: MIT Press, 2001, p. 849–856.

[11] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1978.

[12] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2015.

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[14] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *Journal of Machine Learning Research*, vol. 6, no. 58, pp. 1705–1749, 2005.

[15] S. M. Omohundro, "Five balltree construction algorithms," International Computer Science Institute (ICSI), Berkeley, CA, USA, Tech. Rep. TR-89-063, 1989.

[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, ser. Kdd'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[18] L. S. Shapley, "Notes on the N-Person Game - II: The Value of an N-Person Game," RAND Corporation, Santa Monica, CA, Tech. Rep. RM-670, 1951.

[19] A. E. Roth, Ed., *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, Oct. 1988.

[20] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 4765–4774.

[21] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997.

[22] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, Apr. 1982.

[23] J. E. Matheson and R. L. Winkler, "Scoring Rules for Continuous Probability Distributions," *Management Science*, vol. 22, no. 10, pp. 1087–1096, Jun. 1976.

[24] T. Gneiting, A. E. Raftery, A. H. Westveld, and T. Goldman, "Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation," *Monthly Weather Review*, vol. 133, no. 5, pp. 1098–1118, May 2005.

[25] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics and Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[27] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, ser. Kdd '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794.

[28] P. Bougeault, Z. Toth, C. Bishop, B. Brown, D. Burridge, D. H. Chen, B. Ebert, M. Fuentes, T. M. Hamill, K. Mylne, J. Nicolau, T. Paccagnella, Y.-Y. Park, D. Parsons, B. Raoult, D. Schuster, P. S. Dias, R. Swinbank, Y. Takeuchi, W. Tennant, L. Wilson, and S. Worley, "The THORPEX Interactive Grand Global Ensemble," *Bulletin of the American Meteorological Society*, vol. 91, no. 8, pp. 1059–1072, Aug. 2010.

[29] T. Duan, A. Anand, D. Y. Ding, K. K. Thai, S. Basu, A. Ng, and A. Schuler, "Ngboost: Natural gradient boosting for probabilistic prediction," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 119. Proceedings of Machine Learning Research, 2020, pp. 2690–2700.

[30] Statistisches Bundesamt (Destatis). (2024) Erwerbslose, Erwerbstätige, Erwerbspersonen, Erwerbslosenquote: Deutschland, Monate, Original- und bereinigte Daten. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/268f7f43

[31] Statistisches Bundesamt (Destatis). (2024) Häuserpreisindex: Deutschland, Quartale. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/d17610ea

[32] Statistisches Bundesamt (Destatis). (2024) Beschäftigte, Umsatz im Bauhauptgewerbe: Deutschland, Monate, Wirtschaftszweige. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/8f5178d7

[33] Statistisches Bundesamt (Destatis). (2024) Reallohnindex, Nominallohnindex: Deutschland, Quartale. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/ccdf680d

[34] Statistisches Bundesamt (Destatis). (2024) VGR des Bundes - Konsumausgaben der privaten Haushalte im Inland (nominal/preisbereinigt): Deutschland, Quartale, Original- und bereinigte Daten, Verwendungszwecke. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/54272013

[35] Statistisches Bundesamt (Destatis). (2024) Häuserpreisindex, Preisindex für Bauland: Deutschland, Jahre. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/494e9405

[36] Statistisches Bundesamt (Destatis). (2024) Baugenehmigungen im Hochbau: Deutschland, Jahre, Bautätigkeiten, Gebäudeart. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/46ff7829

[37] Deutsche Bundesbank. (2024) Effektivzinssätze Banken DE / Neugeschäft / Wohnungsbaukredite an private Haushalte, anfängliche Zinsbindung über 10 Jahre / SUD119. [Online]. Available: https://www.bundesbank.de/dynamic/action/de/statistiken/zeitreihen-datenbanken/zeitreihen-datenbank/723452/723452?tsId=BBIM1.M.DE.B.A2C.P.R.A.2250.EUR.N&dateSelect=2024

[38] Statistisches Bundesamt (Destatis). (2024) Vorausberechneter Bevölkerungsstand: Deutschland, Stichtag, Varianten der Bevölkerungsvorausberechnung, Geschlecht, Altersjahre. [On-

line]. Available: https://www-genesis.destatis.de/datenbank/online/url/cd4caafa
[39] Statistisches Bundesamt (Destatis). (2024) Bevölkerung: Deutschland, Stichtag. [Online]. Available: https://www-genesis.destatis.de/datenbank/online/url/08991229
[40] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
[41] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Communications of the ACM*, vol. 29, no. 3, pp. 218–221, 1986.
[42] A. Zimek, E. Schubert, and H. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012.
[43] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," Feb. 2018.