Utilizing TabPFN for Multi-Instance Data with Scarce Labels

Nikolaus Kopp[1], **Alexander Fuchs**[1], **Markus Feuerstein**[2], **Philipp Paller**[2], **Franz Pernkopf**[1]
Christian Doppler Laboratory for Dependable Intelligent Systems in Harsh Environments
SPSC Laboratory - Graz University of Technology[1], RHI Magnesita[2]

Abstract

Tabular data is abundant in critical applications such as science, healthcare, finance, energy, and many other industries, making advances in tabular learning highly influential and interesting for the research community. However, in heavy industry applications we are often presented with a special class of tabular regression problems which are not commonly studied. These multi-instance single-target tabular data problems, originate from the difficulty and cost of taking regular measurements during a production process. In this setting, we have to deal with high-dimensional inputs, in combination with scarce labels. While foundation models such as TabPFN show strong results on suitable datasets, their applicability and performance on multi-instance single-target data is limited by memory and runtime constraints when the number of instances grows. In this paper, we propose a cluster-based dimensionality reduction, which compresses multi-instance measurements by splitting them according to the most relevant cluster constructed from the training set. This approach reduces computational overhead while preserving predictive performance, enabling inference for multi-instance datasets. Our experiments demonstrate that the proposed method extends the practical reach of TabPFN, achieving improved performance across multiple datasets.

1 Introduction

Tabular data remains one of the most common data modalities across science and industry. Although deep learning has achieved breakthroughs in vision and language tasks, its success in tabular problems has been limited, often outperformed by conventional methods such as boosting methods or random forest [Shwartz-Ziv and Armon, 2022]. Recently, the second generation of a foundation model for tabular data, i.e. TabPFNv2, has been introduced [Hollmann et al., 2025], substantially improving on its predecessor TabPFN [Hollmann et al., 2022]. It leverages a vast pretraining corpus of synthetic datasets, i.e. a learned prior, to achieve state-of-the-art performance on tabular data without the need for retraining or fine-tuning.

However, TabPFN was not designed with multi-instance single-target problems in mind, i.e. sequence-to-one tasks, which is a common setting in heavy industry, where taking measurements is challenging and/or expensive. In this setting, the model has to deal with very high-dimensional input data, that can either be reduced via pooling or other methods, or split into a series of independent input-output pairs with duplicated targets. Here, dimensionality reduction is the favored variant, as TabPFN performance has been shown to deteriorate when the training set consists of too many input-output pairs and memory constraints limit the support set size [Zhou et al., 2025]. The issue becomes particularly pronounced in our label-scarce setting, when the number of features far exceeds the number of available targets. The setting of high-dimensional low-sample size datasets is especially relevant in industrial settings where thousands of process parameters are often recorded and only a few hundred data samples are available [Viertauer et al., 2019a,b, Mutsam et al., 2024].

AI for Tabular Data workshop at EurIPS 2025

Since TabPFN can only capture input-output relationships when the effective sample size is on the order of the feature dimension, dimensionality reduction methods, such as hand-crafted descriptors [Fuadah et al., 2024, Zhou et al., 2017], pooling operations [Samih et al., 2023, Meng, 2024], or learned embeddings [Zhou et al., 2025] are commonly applied. While pooling approaches effectively reduce the dimensionality of the data, they are prone to destroy value information. On the other hand, learned embeddings can be more expressive, but require sufficiently many training samples themselves. Therefore, low-data scenarios with high input dimensionality are especially challenging for existing methods.

Our proposed method address these scenarios by employing a cluster-based reduction strategy. To create this cluster-based pooling, we first create cluster instances and then individually assign them to the support- and query set of their respective clusters. Using the proposed method we can improve prediction performance on our industrial steel vessel datasets, as well as benchmark datasets. We compare our method against sum-pooling, Deep Sets [Zaheer et al., 2017] and unsupervised dimensionality reduction using autoencoders.

2 Related work

The introduction of TabPFN by Hollmann et al. [2025] marked an important step for deep learning in tabular data. Its capabilities have been rigorously analyzed [Ye et al., 2025, Zhang et al., 2025], also showing its limitations, including the applicability to small datasets [Zhou et al., 2025]. Several dimensionality or data reduction approaches have been proposed to extend TabPFN to larger datasets, such as combining it with a feature encoder [Liu and Ye, 2025], or selecting support set samples that are closest to the query [Feuer et al., 2024, Koshil et al., 2024]. To directly reduce feature dimensionality, approaches such as feature selection [Hua et al., 2009] and feature transformation have been extensively studied [Ayesha et al., 2020]. Dimensionality reduction can be further enhanced using unsupervised learning methods, such as autoencoder architectures [Mahmud et al., 2021], or neural networks including certain inductive biases as in convolutional neural networks, Behler-Parinello networks, or tailored descriptors [Behler and Parrinello, 2007, Zhou et al., 2017, Fuadah et al., 2024].

3 Methodology

We consider supervised regression settings in which we need to predict a target y_n given a sample consisting of I_n instances $\mathbf{X}_n = \{\mathbf{x}_{n,i}\}_{n=1}^{I_n}$, where $\mathbf{x}_{n,i} \in \mathbb{R}^F$ and F is the number of features per instance. In other words, we want to model a dataset of the form $\mathcal{D} = \{(\mathbf{X}_n, y_n)\}_n^N$, with N being the amount of samples. Additionally, the amount of data per sample greatly exceeds the amount of targets. A simple approach to deal with this high dimensionality is by sum pooling it's instances $\mathbf{X'}_n = \sum_{i=1}^{I_n} \mathbf{x}_{n,i}$, and use them to form support and query set. We improve this method by partitioning the instances into multiple clusters, creating a support and query set for each of them, and aggregating the predictions of TabPFN. This is related to methods studied in literature, where the support set includes samples close to the query set [Feuer et al., 2024]. Note, that in contrast to existing literature, we split samples by partitioning their instances into multiple clusters, and then form, for each cluster, a support and query set using only instances belonging to that cluster.

For simplicity, we will denote the sample indices for training and test using $n \in \{1, 2, \ldots, N_{\mathrm{tr}}\}$ and $n \in \{1, 2, \ldots, N_{\mathrm{te}}\}$, respectively. Context will make the distinction clear.

3.1 K-means pooling

Let' denote the matrix containing all instances of the training set $\mathcal{D}_{\mathrm{tr}}$ as

$$\mathbf{X}_{\mathrm{all}} = \begin{bmatrix} \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{1,I_1} & \cdots & \mathbf{x}_{N_{\mathrm{tr}},1} & \cdots & \mathbf{x}_{N_{\mathrm{tr}},I_{N_{\mathrm{tr}}}} \end{bmatrix}^\top \in \mathbb{R}^{(\sum_{n=1}^{N_{\mathrm{tr}}} I_n) \times F}.$$

We first partition $\mathbf{X}_{\mathrm{all}}$ into K clusters using k-means, then we form a separate support-query set pair for each cluster. Denoting by $\mathcal{I}_{n,k}$ the set of instances of sample n that belong to cluster k, we write the support and query sets as

$$\mathcal{S}_k = \left\{ \left(\frac{I_n}{|\mathcal{I}_{n,k}|} \sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i}, y_n \right) \right\}_{n=1}^{N_{\mathrm{tr}}}, \quad \mathcal{Q}_k = \left\{ \frac{I_n}{|\mathcal{I}_{n,k}|} \sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i} \right\}_{n=1}^{N_{\mathrm{te}}}. \tag{1}$$

Table 1: MAE performance of the TabPFN v2 for all analyzed methods, averaged across 5 folds.

Dataset	Naïve pooling	CAE	Deep Set	KMeans	KMeans-C	KMeans-F	KMeans-FC
Metal	13.5 ± 1.7	14.9 ± 1.9	14.3 ± 1.0	15.3 ± 1.6	$\mathbf{13.4 \pm 1.6}$	15.5 ± 2.9	14.2 ± 3.5
Slag	18.0 ± 1.3	21.3 ± 2.6	19.5 ± 1.6	20.1 ± 1.6	18.0 ± 1.2	21.3 ± 1.2	$\mathbf{17.7 \pm 1.4}$
Bottom in	24.8 ± 1.9	26.4 ± 1.9	27.6 ± 4.0	24.0 ± 2.9	25.3 ± 2.0	$\mathbf{23.0 \pm 2.0}$	$\mathbf{23.0 \pm 1.6}$
Bottom out	28.9 ± 3.5	31.2 ± 4.5	32.6 ± 3.9	30.3 ± 4.8	$\mathbf{28.7 \pm 4.2}$	29.2 ± 5.4	28.8 ± 4.8
M5 food	24.9 ± 5.1	24.7 ± 3.7	26.5 ± 5.2	23.6 ± 3.9	23.6 ± 4.1	23.6 ± 4.4	$\mathbf{23.5 \pm 4.7}$
M5 hobbies	25.6 ± 6.7	25.1 ± 5.6	27.7 ± 7.2	23.6 ± 4.5	$\mathbf{22.6 \pm 4.7}$	24.9 ± 4.9	23.6 ± 4.2
M5 household	24.2 ± 4.5	$\mathbf{23.9 \pm 4.4}$	27.3 ± 2.2	24.0 ± 3.1	$\mathbf{23.9 \pm 3.8}$	26.4 ± 6.5	25.6 ± 6.4

We scale the pooled instances by $I_n/|\mathcal{I}_{n,k}|$ such as the target y_n is the aggregation of all instances. Note that some samples may not be represented in every cluster. Each support–query set pair generates a prediction for every test sample which has at least one instance in the respective cluster. To obtain the final ensemble prediction for a given test sample, we average all query predictions which the sample was part of.

Constructing support–query set pairs as in Equation 1 amounts to describing each sample by multiple selectively pooled representations, one for each cluster. Since the instances of a sample are not distributed uniformly across clusters, these representations may not well describe the sample. Thus we also experiment scaling targets instead of pooled instances to learn the relationship between instances and their contribution to the target. The support and query set are constructed as

$$\mathcal{S}_k = \left\{ \left(\sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i}, \frac{|\mathcal{I}_{n,k}|}{I_n} y_n \right) \right\}_{n=1}^{N_{\mathrm{tr}}}, \quad \mathcal{Q}_k = \left\{ \sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i} \right\}_{n=1}^{N_{\mathrm{te}}}. \tag{2}$$

Since each support-query set pair now predicts only a fractional contribution to the target, the final ensemble prediction for a test sample is obtained by summing over the individual predictions.

Given that pooling reduces each sample to an F-dimensional vector, and F is typically smaller than the number of training samples N_{tr}, we have the additional option to concatenate the selectively pooled instances with the pooled instances (all instances of sample) for both the support- and query sets. This gives us four methods: *KMeans* and *KMeans-F* use Equations 1 and 2 for support-query set pair construction, respectively, and *KMeans-C* and *KMeans-FC* are the respective variants, where the feature space is extended by the pooled instances. Aggregation details are explained in Appendix B.

4 Experimental setup

We use four industrial datasets from steel-processing [Mutsam et al., 2024]. To expand our analysis, we adapt the public dataset M5 [Howard et al., 2020] to our multiple-instance setting, creating 3 separate datasets. Dataset details and how they were created is summarized in Appendix A. All methods are evaluated using 5-fold cross-validation with a shared set of folds. Regarding M5 datasets, which exhibit temporal and seasonal patterns, each fold consists of consecutive samples to preserve the temporal structure.

We compare the TabPFN v2 performance for our dimensionality reduction method against naïve pooling (pooling all instances), a convolutional autoencoder-based (CAE-based) preprocessing model, and against Deep Set regressors [Zaheer et al., 2017]. We performed a hyperparameter search for CAEs and Deep Sets, where we tested different regularization strengths, and our k-means based method, where we varied the number of clusters. We optimized hyperparameters for each training fold separately. Both the autoencoders and Deep Set networks were trained with AdamW, using early stopping.

5 Results

Table 1 shows the TabPFN v2 performance of all methods across datasets, we see that all methods exhibit high variance between folds, limiting the reliability of comparisons. This variability arises from small sample sizes, causing substantial shifts in the sample distribution across folds. Nonetheless, some trends can be observed. The Deep Sets model and the CAE are generally outperformed by

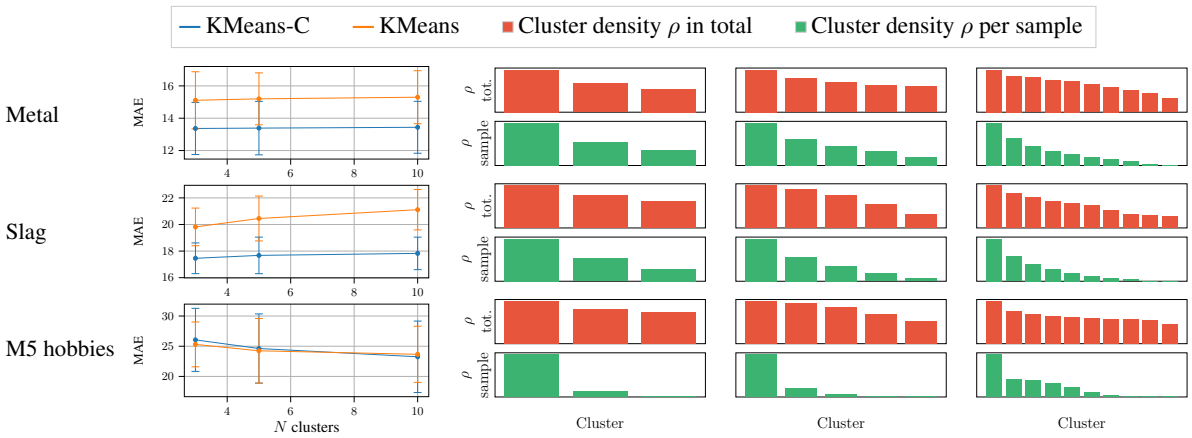

Figure 1: KMeans and KMeans-C applied to the three datasets: Metal, Slag, and M5 hobbies. On the left, the MAE is plotted over the number of used clusters. The three right column show the corresponding statistics of cluster assignments. Cluster density ρ in total shows the distribution across clusters of all instances in the training sets, displayed in descending order. Cluster density ρ per sample shows the average assignment distribution of instances within a sample.

simpler pooling strategies. While KMeans tends to perform worse than naïve pooling on industrial datasets, it improves results on the M5 datasets. This suggests that its clustering-based instance selection may be more effective if the underlying data distribution is more structured and less noisy. KMeans-C, in contrast, benefits from combining the two approaches, consistently achieving results that are comparable or better than those of naive pooling and standard KMeans. Interestingly, KMeans-F and KMeans-FC generally perform worse than their respective counterparts, despite their invariance to the fraction of instances assigned to each cluster.

Figure 1 shows an ablation study of KMeans and KMeans-C conducted on three datasets. On the industrial datasets we observe that performance does not improve with more clusters. We argue that these datasets are relatively noisy, where using a fine-grained clustering becomes unreliable. Additionally, the spread in MAE between KMeans-C and KMeans is high on industrial datasets, while it is low on M5 hobbies (a dataset on which it performs well). This suggests that KMeans-C only works well on industrial datasets because of the access to sum-pooled instances. Furthermore, we see that the entropy of sample-wise cluster assignments is lower, while the entropy of total cluster assignments is higher for M5 compared to the industrial datasets. This indicates that many samples of the M5 datasets contain instances that belong to a small number of clusters, which may be critical for improved performance. We argue that the more deterministic the sample-wise cluster assignments become, the more our method resembles context optimization techniques from literature [Feuer et al., 2024]. We thus suspect, that the limited performance on industrial data could originate from insufficient separation of instances, or samples, when using the k-means–based methods.

Overall, our findings suggest that ensemble predictions based on selective pooling can boost performance. However, KMeans-C's gains may be partly attributed to ensemble prediction and an expanded feature space (see Appendix C). It is conceivable, that the k-means algorithm is not be the best suited method for this task. Generally, our analysis is limited by few benchmarks and results with high variance.

6 Conclusion

We introduced a method to reduce the dimensionality of multi-instance data by clustering and constructing support-query pairs for each cluster, enabling predictions with TabPFN v2. Our approach shows promising results, though for now the analysis is limited by few datasets and high variance. Future work could explore alternative clustering strategies, analyze dropping problematic clusters, and extend the method to categorical features. In particular, while our approach relies on Euclidean distance in feature space, future work may explore more suitable similarity measures or clustering on transformed representations, as well as richer, ways of summarizing clusters beyond average pooling.

Acknowledgments and Disclosure of Funding

The financial support by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged. Furthermore, the research was funded by RHI Magnesita.

References

Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58, 2020.

Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

Benjamin Feuer, Robin T Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024.

Yunendah Nur Fuadah, Muhammad Adnan Pramudito, Lulu Firdaus, Frederique J Vanheusden, and Ki Moo Lim. Qsar classification modeling using machine learning with a consensus-based approach for multivariate chemical hazard end points. *ACS omega*, 9(51):50796–50808, 2024.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL `https://www.nature.com/articles/s41586-024-08328-6`.

Addison Howard, inversion, Spyros Makridakis, and vangelis. M5 forecasting - accuracy. `https://kaggle.com/competitions/m5-forecasting-accuracy`, 2020.

Jianping Hua, Waibhav D Tembe, and Edward R Dougherty. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409–424, 2009.

Mykhailo Koshil, Thomas Nagler, Matthias Feurer, and Katharina Eggensperger. Towards localization via data embedding for tabpfn. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024.

Si-Yang Liu and Han-Jia Ye. Tabpfn unleashed: A scalable and effective solution to tabular classification problems, 2025. URL `https://arxiv.org/abs/2502.02527`.

Mohammad Sultan Mahmud, Joshua Zhexue Huang, Xianghua Fu, Rukhsana Ruby, and Kaishun Wu. Unsupervised adaptation for high-dimensional with limited-sample data classification using variational autoencoder. *Computing and informatics*, 40(1):1–28, 2021.

Yigang Meng. Music genre classification: A comparative analysis of cnn and xgboost approaches with mel-frequency cepstral coefficients and mel spectrograms. In *AIP Conference Proceedings*, volume 3194, page 020002. AIP Publishing LLC, 2024.

Nikolaus Mutsam, Alexander Fuchs, Fabio Ziegler, and Franz Pernkopf. Data-scarce condition modeling requires model-based prior regularization. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6695–6699, 2024. doi: 10.1109/ICASSP48485.2024.10446987.

Amina Samih, Abderrahim Ghadi, and Abdelhadi Fennan. Enhanced sentiment analysis based on improved word embeddings and xgboost. *International Journal of Electrical and Computer Engineering*, 13(2):1827–1836, 2023.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Andreas Viertauer, Nikolaus Mutsam, Franz Pernkopf, Andreas Gantner, Georg Grimm, G Lammer, and Alexander Ratz. Refractory lifetime prognosis for RH-degassers. In *UNITECR*, 2019a.

Andreas Viertauer, Nikolaus Mutsam, Franz Pernkopf, Andreas Gantner, Georg Grimm, Waltraud Winkler, Gregor Lammer, and Alexander Ratz. Refractory condition monitoring an dlifetime prognosis for rh degasser. In *AISTech – The Iron & Steel Technology Conference and Exposition*, 2019b.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at tabpfn v2: Strength, limitation, and extension. *arXiv preprint arXiv:2502.17361*, 2025.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Qiong Zhang, Yan Shuo Tan, Qinglong Tian, and Pengfei Li. Tabpfn: One model to rule them all? *arXiv preprint arXiv:2505.20003*, 2025.

Chang Zhou, Hua Yu, Yijie Ding, Fei Guo, and Xiu-Jun Gong. Multi-scale encoding of amino acid sequences for predicting protein interactions using gradient boosting decision tree. *PLoS One*, 12 (8):e0181426, 2017.

Summer Zhou, Vinayak Agarwal, Ashwin Gopinath, and Timothy Kassis. The limitations of tabpfn for high-dimensional rna-seq analysis. *bioRxiv*, pages 2025–08, 2025.

Table 2: Statistics of the datasets, including number of samples N, number of features F and the average number of instances per samples $\bar{I}_n$.

Dataset	N	F	$\bar{I}_n$
Metal	230	15	113.8
Slag	449	20	58.8
Bottom in	164	40	160.6
Bottom out	166	40	160.6
M5 food	163	25	70.3
M5 hobbies	156	25	73.5
M5 household	161	25	71.3

A Datasets

Each industrial dataset consists of samples representing a vessel's usage period, called *campaigns*. A campaign consists of multiple operational steps (instances). The goal is to predict the refractory wear at the end of a vessel's usage period, given feature vectors for each step. This naturally creates a multiple-instance learning scenario, with the aggregated wear over all instances as target.

We further adapted the M5 dataset to our multi-instance settings. Each instance corresponds to a single product sold at a particular store, characterized by features including a cosine–sine encoding of the month, seven binary weekday indicators, four binary event-type indicators, a one-hot encoding of the store, a binary SNAP flag, and the store's sell price. For each product category (food, household, hobbies), we randomly select one product to form three datasets. Samples are constructed by partitioning the time series into intervals of 7–15 days. For each interval, we create one sample containing all instances from six randomly selected stores. The target associated with each sample is the total sales across these stores during that interval.

Features were first scaled to a normal distribution by a quantile transformation, fitted to instances of the training set $\{\mathbf{x}_{n,i}\}_{n,i}$. Table 2 shows some statistics of all datasets.

B Support set construction

In this section, we clarify how we create the support set for each method. Note that the query set is constructed analogously, except that the target is omitted. Whenever we refer to pooling instances without using the term *selective*, we mean pooling over all instances belonging to a sample. In contrast, *selective* pooling refers to pooling only over instances that belong to a specific cluster.

B.1 KMeans

Here we scale the selectively pooled instances such that they preserve information about the total target magnitude (see also Equation 1:

$$\mathcal{S}_k = \left\{ \left(\frac{I_n}{|\mathcal{I}_{n,k}|} \sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i},\, y_n \right) \right\}_{n=1}^{N_{\text{tr}}}. \tag{3}$$

B.2 KMeans-F

With KMeans-F, we attempt to learn the fractional contribution of selected instances to the target. Thus, we sum-pool the selected instances and scale the target by the fraction of instances belonging to the respective cluster (see also Equation 2:

$$\mathcal{S}_k = \left\{ \left(\sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i},\, \frac{|\mathcal{I}_{n,k}|}{I_n} y_n \right) \right\}_{n=1}^{N_{\text{tr}}}. \tag{4}$$

B.3 KMeans-C

When concatenating selectively pooled instances with sum-pooled instances, we no longer need to scale the former. Thus, the support set becomes

$$\mathcal{S}_k = \left\{ \left(\Big[\frac{1}{|\mathcal{I}_{n,k}|} \sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i} \,;\, \sum_{i=1}^{I_n} \mathbf{x}_{n,i} \Big],\, y_n \right) \right\}_{n=1}^{N_{\text{tr}}}, \tag{5}$$

where $[\cdot\,;\,\cdot]$ denotes concatenation.

B.4 KMeans-FC

Since the goal of KMeans-F is to model the fractional target contribution, concatenating with the sum-pooled representation is not meaningful. To see this, consider augmenting a sample by adding more instances that do not belong to a given cluster—this should not affect the corresponding support–query pair. Therefore, we construct the support set as

$$\mathcal{S}_k = \left\{ \left(\Big[\sum_{i \in \mathcal{I}_{n,k}} \mathbf{x}_{n,i} \,;\, \frac{1}{I_n} \sum_{i=1}^{I_n} \mathbf{x}_{n,i} \Big],\, \frac{|\mathcal{I}_{n,k}|}{I_n} y_n \right) \right\}_{n=1}^{N_{\text{tr}}}. \tag{6}$$

C Additional baseline comparisons

In this section we want to expand the baseline comparisons. To this end, we study four more baselines, two of which serve as dummies. *DummyRegressor* always predicts the mean target value found in the training set and *LinearRegressor* is trained by receiving only the number of instances as input. LinearRegressor can be understood as a dummy regressor, when the target value is decorrelated from the number of instances I_n. We further also introduce a randomized version of KMeans and KMeans-C, which we call RCluster and RCluster-C, respectively. These methods are similar with respect to their counterparts, with the only change that they assign instances to clusters randomly. Table 3 shows the results of these baselines together with KMeans and KMeans-C.

Our main interest lies in comparing KMeans-C to RCluster-C to assess whether KMeans-C's improvements stem primarily from its expanded feature space and ensemble predictions. Indeed, RCluster-C performs better than naive pooling and even surpasses KMeans-C on some datasets. We speculate that this phenomenon occurs, because RCluster-C does something similar to data augmentation: By training multiple predictors on different training sets, the effective training set size is increased. Additionally, averaging over many randomly selected samples resembles noisy average-pooling.

The results raise the question whether clustering itself matters at all. As shown in Table 3, RCluster-C's mean MAE differs from KMeans-C's by at most 0.5 on most datasets, though it performs worse by 1.3 and 2.4 points on two others. Though the high variance prevents reliable conclusions, it appears that k-means can indeed help on specific datasets and that its advantage on others could largely stem from indirect data augmentation effects. Figure 1 also shows that clustering behaves differently on M5 hobbies, a dataset where KMeans-C is strong, than on Metal and Slag: First, on M5 hobbies, KMeans-C gains performance as the number of clusters increases. Second, the MAE achieved by KMeans and KMeans-C shows very little spread - in stark contrast to the industrial datasets. Third, instances are more uniformly distributed across clusters, whereas the instances belonging to a given sample tend to fall into fewer clusters. It is conceivable that clustering in feature space does not yield optimal separation. Further, using Euclidean distance with mean or sum aggregation might fail to capture instance-level structure effectively. Extensive experiments are needed to clarify this.

Table 3: Mean and standard deviation of the MAE for each method, evaluated using 5-fold cross-validation.

Dataset	DummyRegressor	LinearRegressor	RCluster	RCluster-C	KMeans	KMeans-C
Metal	16.9 ± 0.9	15.5 ± 2.2	14.3 ± 2.0	13.5 ± 1.6	15.3 ± 1.6	13.4 ± 1.6
Slag	23.8 ± 1.5	23.3 ± 1.3	19.5 ± 1.6	18.1 ± 1.1	20.1 ± 1.6	18.0 ± 1.2
Bottom in	30.1 ± 2.6	28.6 ± 2.7	24.8 ± 1.8	24.8 ± 1.8	24.0 ± 2.9	25.3 ± 2.0
Bottom out	32.0 ± 5.3	30.0 ± 6.5	29.7 ± 3.8	28.4 ± 3.4	30.3 ± 4.8	28.7 ± 4.2
M5 food	37.4 ± 3.5	25.5 ± 5.5	24.3 ± 4.4	24.9 ± 4.8	23.6 ± 3.9	23.6 ± 4.0
M5 hobbies	41.4 ± 6.9	29.7 ± 6.1	23.9 ± 5.2	25.0 ± 6.3	23.6 ± 4.5	22.6 ± 4.7
M5 household	36.8 ± 7.1	27.5 ± 3.3	23.6 ± 3.1	23.5 ± 4.5	24.0 ± 3.1	23.9 ± 3.8