Comparing Task-Agnostic Embedding Models for Tabular Data

Frederik Hoppe[*]
CONTACT Software GmbH
Bremen, Germany

Lars Kleinemeier[*]
CONTACT Software GmbH
Bremen, Germany

Astrid Franz
CONTACT Software GmbH
Bremen, Germany

Udo Göbel
CONTACT Software GmbH
Bremen, Germany

Abstract

Recent foundation models for tabular data achieve strong task-specific performance via in-context learning. Nevertheless, they focus on direct prediction by encapsulating both representation learning and task-specific inference inside a single, resource-intensive network. This work specifically focuses on representation learning, i.e., on transferable, task-agnostic embeddings. We systematically evaluate task-agnostic representations from tabular foundation models (TabPFN and TabICL) alongside with classical feature engineering (TableVectorizer) across a variety of application tasks as outlier detection (ADBench) and supervised learning (TabArena Lite). We find that simple TableVectorizer features achieve comparable or superior performance while being up to three orders of magnitude faster than tabular foundation models. The code is available at https://github.com/ContactSoftwareAI/TabEmbedBench.

1 Introduction

Recent foundation models for tabular data, such as TabPFN [5], TabICL [7], ConTextTab [9], and TabStar [1], leverage in-context learning to achieve strong task-specific performance on structured datasets. These models are trained on a huge number of synthetic datasets to capture underlying patterns in tabular data, enabling them to adapt quickly to new datasets through contextual information. However, unlike large language models (LLMs) that produce versatile, task-agnostic embeddings useful across diverse downstream applications - such as similarity search, retrieval, clustering, and few-shot transfer - these tabular models focus primarily on direct prediction rather than generating transferable representations.

Despite the significant progress in task-specific tabular learning, there remains a substantial gap in exploring task-agnostic embeddings for tabular data. From both research and industrial perspectives, such embeddings offer valuable flexibility by producing representations that can be reused across multiple tasks - including outlier detection, classification, and regression - without task-specific customization or retraining. Yet, to the best of the authors' knowledge, there has not been a comprehensive evaluation comparing task-agnostic embedding models for tabular data.

This paper addresses this gap by systematically evaluating embeddings extracted from foundation models like TabPFN and TabICL as task-agnostic representations of individual data samples (table rows), enabling tasks like similarity search, outlier detection, regression and classification on a single table. They are a step towards applications such as cross-dataset search in data lakes, entity resolution,

[*]{frederik.hoppe, lars.kleinemeier}@contact-software.com

AI for Tabular Data workshop at EurIPS 2025

and comparing records across different sources. Therefore, this study focuses on row embeddings since individual records are the basic unit for most machine learning tasks, unlike column or table embeddings. Our primary contribution is to initiate discussion about the potential of general-purpose tabular embeddings, drawing parallels to the success of LLM embeddings. By highlighting both promise and current limitations we aim to establish task-agnostic tabular embeddings as a valuable research direction for the community.

2 Embedding Models

Learning effective task-agnostic representations of tabular data is challenging due to its heterogeneous nature, combining numerical and categorical features with complex dependencies that vary across datasets. While this initial study focuses on numerical and categorical tabular data, further research should incorporate other modalities such as text, see Section 5. We employ three approaches to obtain per-row tabular embeddings, spanning classical feature engineering to deep learned embeddings: TableVectorizer for statistical encoding, TabICL and TabPFN as tabular foundation models pre-trained on specific tasks yet capable of generating task-agnostic row embeddings.

TableVectorizer. TableVectorizer from the skrub package [10] serves as a preprocessing baseline that converts tabular data into numerical representations through adaptive column-wise encoding, automatically selecting appropriate encoders (one-hot encoding, datetime decomposition, or dimensionality-reduced string encoding) based on each column's characteristics before concatenating features into row embeddings. It is a non-learned, unsupervised method whose embedding dimension depends on the dataset characteristics (ranging from 4 to 166 in our benchmarks).

TabICL. TabICL [7] is a tabular foundation model that uses In-Context Learning for classification without parameter updates. The embeddings are extracted from the first stage, where a Set Transformer processes columns independently to create distribution-aware features. This is followed by a transformer with rotary positional embeddings that models inter-feature dependencies row-by-row. This process produces semantically rich representations via learnable [CLS] tokens. The embedding dimension is 512.

TabPFN. TabPFN [5], an additional foundation model that has been pre-trained on millions of synthetic datasets, employs a two-way attention mechanism that alternates between features and samples. Following the tabpfn-extensions approach [11], we generate embeddings through feature masking. In this process, each feature is iteratively predicted from the remaining features using the TabPFNClassifier or TabPFNRegressor. We then extract the representations of each feature that are aggregated into final row embeddings by taking the average per component. The embedding dimension is 192.

3 Experiments

We evaluate embeddings on two benchmark suites: ADBench [4] for unsupervised outlier detection tasks and TabArena [3] for supervised classification and regression tasks. From ADBench, we select only datasets labeled as tabular while excluding those from the image category, resulting in a curated collection of real-world tabular datasets. For TabArena, we use the TabArena Lite preset, which evaluates only the first fold of the first repeat for all tasks. This lightweight configuration reduces computational requirements while maintaining dataset diversity across 51 manually curated OpenML datasets. We impose upper bounds of 10,000 samples and 200 features per dataset to keep the computation time in a reasonable range. This results in 27 datasets each for ADBench and TabArena.

For each dataset, we generate embeddings and evaluate them using lightweight predictive algorithms (K-Nearest Neighbors, Multi-Layer Perceptrons with Optuna-based hyperparameter optimization [12]), and specialized outlier detection algorithms (Local Outlier Factor (LOF) [2], Isolation Forest (IF) [6], and DeepSVDD [8]), details can be found in Appendix A. Following the respective benchmark protocols, TabArena Lite uses stratified train/test splits with single-fold evaluation, while ADBench outlier detection employs the full dataset approach.

As evaluation measures, we use the area under the receiver operating characteristic curve (AUC) for outlier detection and classification and the mean absolute percentage error (MAPE) for regression.

The AUC score ranges from 0 to 1 and should be as large as possible, whereas the MAPE score is non-negative and should be as small as possible.

4 Results and Discussion

The following discussion focuses on task-agnostic embeddings. Tabular foundation models such as TabPFN and TabICL are primarily constructed for prediction leveraging in-context learning. However, they also provide promising task-independent embeddings. Whenever we refer to TabPFN or TabICL, we mean the task-agnostic embeddings constructed by these models.

In the case of outlier detection, Table 1 shows that TableVectorizer embeddings outperform the ones of TabICL and TabPFN for every outlier algorithm. For LOF and IF, the task-agnostic embeddings of TabICL achieve better results than the ones of TabPFN, while the embeddings of the latter yield a higher AUC score for DeepSVDD and its dynamical variant. For the supervised tasks shown in Tables 2, 3 and 4, TabICL performs best. Since the number of suited TabArena datasets was limited at the time of this investigation, general ranking conclusions for the three investigated embedding models cannot be drawn from these results. Nevertheless, for the used datasets it can be said, that TableVectorizer as simple feature engineering method keeps up or even outperforms tabular foundation models like TabICL and TabPFN when considering task-agnostic embeddings. Additionally, TableVectorizer constructs embeddings with the lowest dimension ($\leq$ 166), while TabPFN requires 192 and TabICL even 512 as embedding dimension.

Table 1: Performance for outlier detection (AUC score $\pm$ standard deviation) for the three embedding models. The scores are averaged over 27 datasets.

Embedding Model	Time (s)	LOF	IF	DeepSVDD	DeepSVDD-dyn
TabICL	0.086	0.678±0.193	0.740±0.190	0.633±0.172	0.609±0.199
TabPFN	75.839	0.675±0.171	0.693±0.205	0.654±0.195	0.642±0.210
TableVectorizer	0.004	0.717±0.182	0.764±0.197	0.683±0.210	0.683±0.210

Table 2: Performance for binary classification (AUC score $\pm$ standard deviation) for the three embedding models. The scores are averaged over 19 datasets.

Embedding Model	Time (s)	KNNClassifier	MLPClassifier
TabICL	0.115	0.810±0.094	0.830±0.096
TabPFN	294.234	0.749±0.098	0.786±0.093
TableVectorizer	0.012	0.739±0.111	0.827±0.094

Table 3: Performance for multiclass classification (AUC score $\pm$ standard deviation) for the three embedding models. The scores are averaged over 6 datasets.

Embedding Model	Time (s)	KNNClassifier	MLPClassifier
TabICL	0.067	0.901±0.115	0.927±0.067
TabPFN	146.677	0.854±0.109	0.889±0.082
TableVectorizer	0.014	0.851±0.129	0.905±0.071

Table 4: Performance for regression (MAPE score $\pm$ standard deviation) for the three embedding models. The scores are averaged over 7 datasets.

Embedding Model	Time (s)	KNNRegressor	MLPRegressor
TabICL	0.063	0.257±0.277	0.151±0.114
TabPFN	24.162	0.322±0.334	0.347±0.292
TableVectorizer	0.008	0.308±0.403	0.162±0.132

Beyond its performance, TableVectorizer offers a notable advantage in terms of embedding computation efficiency, requiring significantly less time than TabICL and TabPFN due to their more sophisticated model architecture. In particular, TabPFN requires a high amount of time since target-aware embeddings are computed for every column and are aggregated afterwards. In contrast, TabICL does not require customization as it naturally constructs task-agnostic embeddings, which are further processed by an in-context learning block when used directly for prediction.

A significant limitation of TableVectorizer is its variable embedding dimensionality, which is determined by the input dataset. This dataset-dependent dimensionality poses a challenge for applications requiring the joint embedding of diverse datasets into a unified representation space, which is a widespread scenario in industry. In contrast, models like TabPFN and TabICL employ a fixed, but significantly larger embedding size across all datasets.

5 Conclusion

This brief study demonstrates that simple feature engineering methods like TableVectorizer achieve competitive performance for task-agnostic embeddings while requiring substantially fewer computational resources in settings where sophisticated tabular foundation models cannot fully leverage their in-context learning capabilities. However, several open challenges remain for future work. First of all, our findings have to be confirmed on a wider range of datasets. Additionally, more embedding models should be included in this comparison. Although the embeddings are constructed in a task-independent way, to assess their *task-agnostic* quality, additional downstream tasks and metrics should evaluate the embeddings.

The ingestion of new rows raises fundamental questions about context selection for in-context learners: should entire datasets be re-embedded, or can a representative subset suffice? Distribution shift poses additional complications: If new data exhibit covariate shift or label drift, in-context models may produce embeddings misaligned with the original space.

Beyond these practical concerns, current methods primarily capture statistical patterns but lack semantic awareness. Recent works such as ConTextTab [9] and TabSTAR [1] demonstrate promising directions by incorporating textual metadata and target-aware representations, suggesting pathways towards semantically richer embeddings. Extending these approaches to relational databases represents a natural progression, where tabular foundation models could serve as building blocks for relational foundation models that learn representations across joined tables and foreign-key relationships.

Shared representation spaces across diverse datasets also warrant investigation: do models sufficiently differentiate between datasets when embedded jointly, or does cross-dataset contamination occur? Finally, assuming that semantically aware embeddings can be achieved, multimodal alignment becomes feasible, enabling tabular representations to be aligned with text, images, or other modalities for cross-modal retrieval and reasoning. Addressing these challenges will be essential for establishing task-agnostic tabular embeddings as a flexible, reusable primitive in the broader machine learning ecosystem.

References

[1] Alan Arazi, Eilam Shapira, and Roi Reichart. TabSTAR: A foundation tabular model with semantically target-aware representations. https://arxiv.org/abs/2505.18125, 2025.

[2] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, page 93–104, New York, NY, USA, 2000.

[3] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. https://arxiv.org/abs/2506.16791, 2025.

[4] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 32142–32159, 2022.

[5] 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*, 637:319–326, 2025.

[6] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1):1–39, 2012.

[7] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, Vancouver, Canada, 2025.

[8] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402, 2018.

[9] Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. ConTextTab: A semantics-aware tabular in-context learner. In *Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data*, Vancouver, Canada, 2025.

[10] https://skrub-data.org/, https://github.com/skrub-data/skrub/.

[11] https://github.com/PriorLabs/tabpfn-extensions/.

[12] https://optuna.org/.

A Experimental Details

All evaluators inherit from an abstract base class ensuring consistent training and prediction workflows. Below we describe the specific algorithms and their configurations.

A.1 Unsupervised Task: Outlier Detection

Local Outlier Factor (LOF). We utilize scikit-learn's `LocalOutlierFactor` implementation, which computes local density deviations by comparing each sample's density to its neighbors' densities. Higher scores indicate a greater likelihood of outliers. Since the number of neighbors, `n_neighbors`, is the only parameter influencing the result, we iterate over this parameter, i.e., `n_neighbors` $\in \{5,10,15,20,25,30,35,40,45\}$, and record the optimal score per embedding model and dataset for evaluation.

Isolation Forest (IF). We employ scikit-learn's `IsolationForest`, which isolates anomalies through recursive random feature partitioning. Outliers require fewer splits for isolation, yielding higher anomaly scores. We iterate over the number of estimators, i.e., `n_estimators` $\in \{50,100,150,200,250\}$, and record the optimal score per embedding model and dataset for evaluation.

Deep Support Vector Data Description (DeepSVDD). We use PyOD's DeepSVDD implementation in two configurations:

1. Default parameters from the package with fixed hidden layer architecture (64, 32), and
2. Dynamically adjusted hidden layers for higher-dimensional embeddings, where layer sizes are computed as decreasing powers of 2 starting from approximately half the embedding dimension (if at least larger than 64) down to 32.

Both variants employ the default parameters from the PyOD package for the remaining parameters, i.e., ReLU activation for the hidden layers, Sigmoid activation for the output layer, dropout rate of 0.2 and ℓ_2-regularization with strength 0.1, with training over 200 epochs using batch size 32. The neural network attempts to map most of the embeddings into a hypersphere. Normal data points are mapped to the interior of the hypersphere, whereas anomalous points are mapped to the exterior.

A.2 Supervised Tasks: Classification and Regression

K-Nearest Neighbors (KNN). We employ scikit-learn's `KNeighborsClassifier` and `KNeighborsRegressor` with distance metrics euclidean and distance-based weighting scheme. For classification tasks, the model returns class probability predictions; for regression, it outputs continuous values. We iterate over the number of neighbors, i.e., `n_neighbors` $\in \{5,10,15,20,25,30,35,40,45\}$, and record the optimal score per embedding model and dataset for evaluation.

Multi-Layer Perceptron (MLP). We implement a PyTorch-based MLP with Optuna hyperparameter optimization using Tree-structured Parzen Estimator (TPE) sampling. The search space encompasses: number of hidden layers (1-5 for classification, 1-3 for regression), hidden dimensions (32-512, log-scaled), dropout rates (0.0-0.5), learning rates (0.0001 to 0.01, log-scaled), batch sizes 16, 32, 64, 128, and training epochs (50-200). Models employ StandardScaler normalization, Adam optimization, and early stopping with 10-epoch patience. We conduct 50 optimization trials with 5-fold cross-validation.

B Detailed Results

In the following, we include tables and figures that show the results in more detail.

B.1 Outlier Detection

The AUC distribution for the three investigated embedding models and the different outlier detection algorithms are shown in Figure 1. The wide range of AUC values indicates that the difficulty of the investigated datasets strongly varies. The outlier ratio of the chosen 27 ADBench datasets ranges from 1% to 40%. Table 5 shows a breakdown of the AUC results with respect to the outlier ratio in the datasets. As can be seen, the performance deteriorates with increasing outlier ratio. The variance between the three investigated embedding models is most pronounced for small outlier ratios, it vanishes for large outlier ratios.

Table 5: Performance for outlier detection (AUC score $\pm$ standard deviation) for the three embedding models, the chosen ADBench datasets are grouped by outlier ratio. The scores are averaged over the algorithms (DeepSVDD, DeepSVDD-dynamic, IsolationForest, LocalOutlierFactor) and datasets.

Outlier Ratio	#Datasets	TabICL	TabPFN	TableVectorizer
$< 5\%$	9	0.776±0.213	0.817±0.216	0.841±0.184
5-10%	6	0.657±0.181	0.660±0.146	0.769±0.173
$\geq 10\%$	12	0.585±0.137	0.555±0.102	0.586±0.144

B.2 Classification and Regression

The AUC distribution for binary and multiclass classification and the MAPE distribution for regression for the three investigated embedding models and the two different algorithms (KNN, MLP) are shown in Figures 2, 3 and 4. Since the number of suited TabArena datasets was limited at the time of this investigation, boxplots showing the distribution of the results may not be expressive, especially for multiclass classification and regression, where only 6 datasets where available. As a continuously updated benchmark for machine learning on tabular data, TabArena is expected to expand its repository with increasingly appropriate datasets.

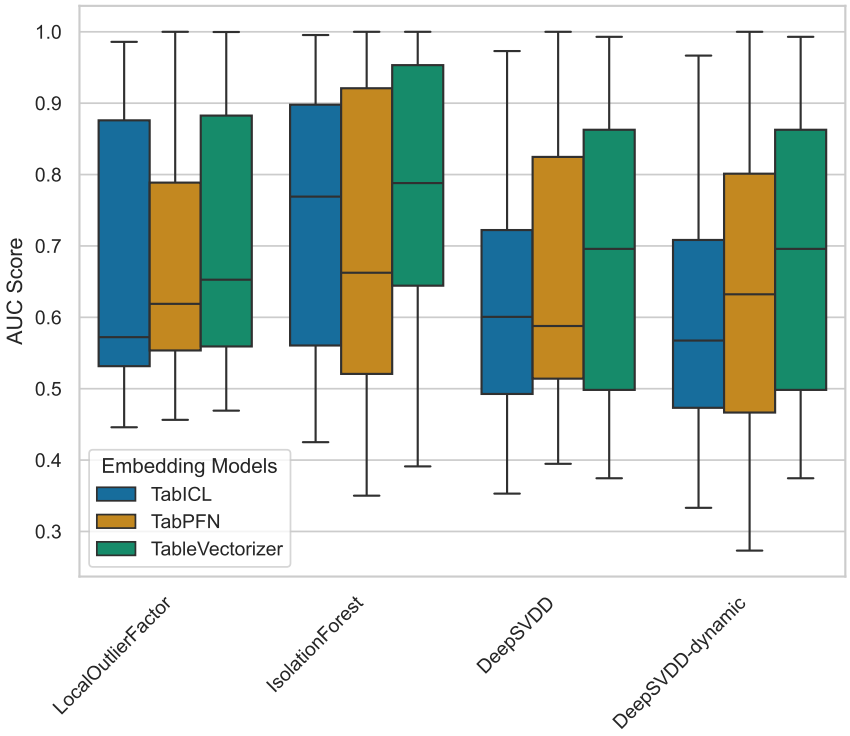

Figure 1: Distribution of the performance for outlier detection (AUC) for the three embedding models shown in Table 1.

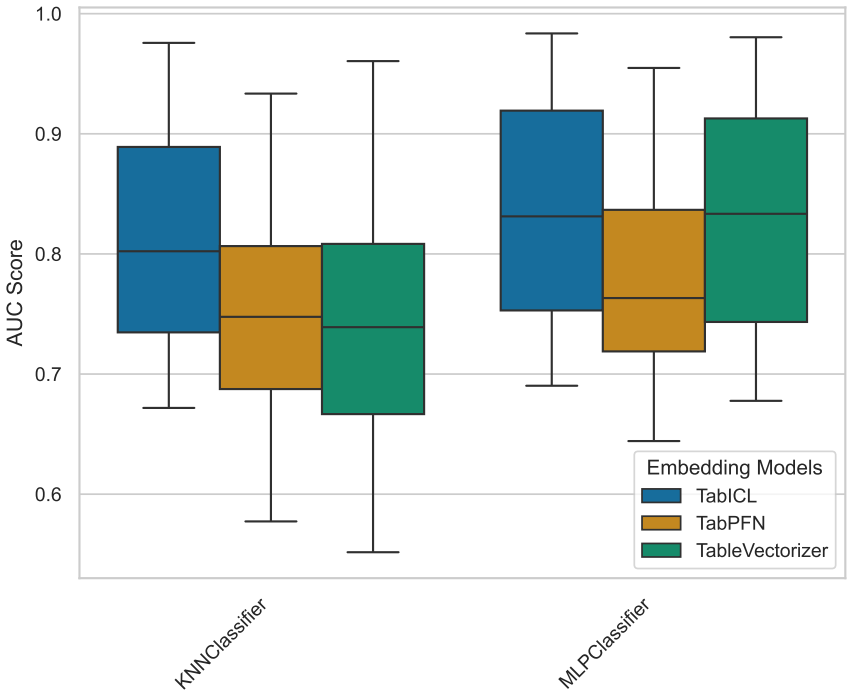

Figure 2: Distribution of the performance for binary classification (AUC) for the three embedding models shown in Table 2.

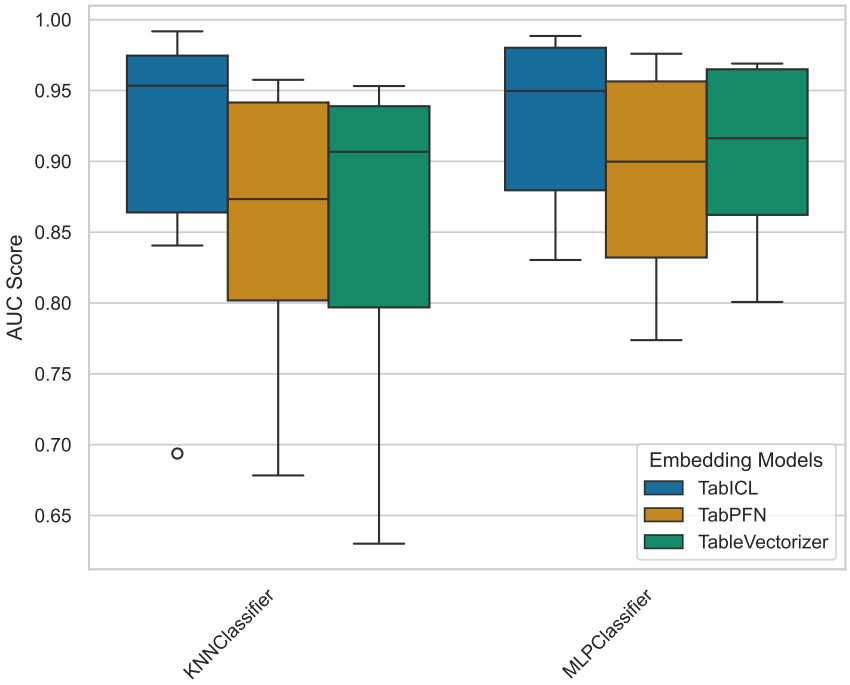

Figure 3: Distribution of the performance for multiclass classification (AUC) for the three embedding models shown in Table 3.

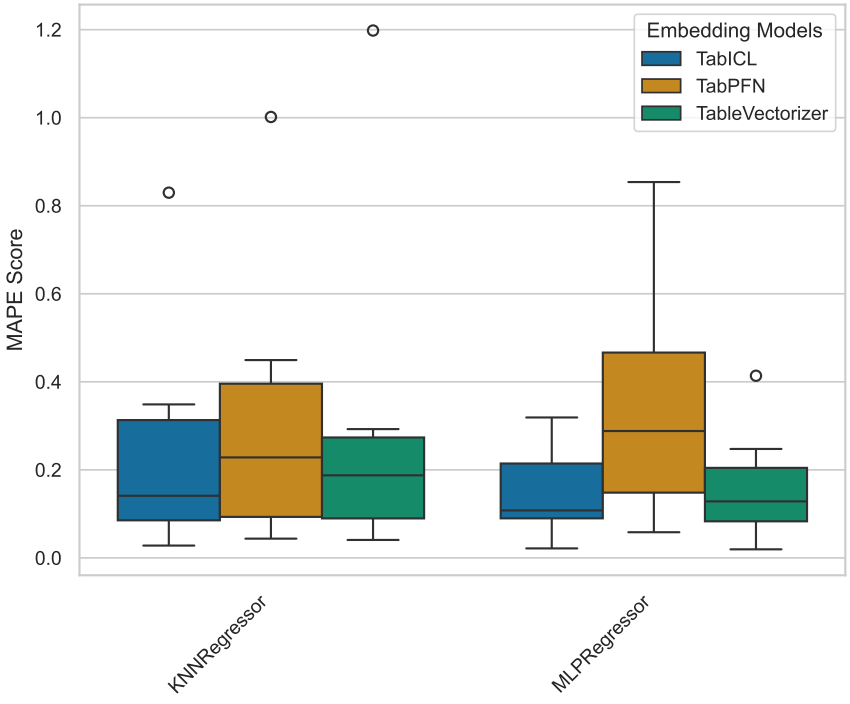

Figure 4: Distribution of the performance for regression (MAPE) for the three embedding models shown in Table 4.