Analytics Modelling over Multiple Datasets Using Vector Embeddings

Andreas Loizou(✉) and Dimitrios Tsoumakos

Database and Knowledge Systems Lab, School of ECE, National Technical University of Athens, Athens, Greece
{antreasloizou,dtsouma}@mail.ntua.gr

Abstract. The massive increase in the data volume and dataset availability for analysts compels researchers to focus on data content and select high-quality datasets to enhance the performance of analytics operators. While selecting high-quality data significantly boosts analytical accuracy and efficiency, the exact process is very challenging given large-scale dataset availability. To address this issue, we propose a novel methodology that infers the outcome of analytics operators by creating a model from the available datasets. Each dataset is transformed to a vector embedding representation generated by our proposed deep learning model *NumTabData2Vec*, where similarity search are employed. Through experimental evaluation, we compare the prediction performance and the execution time of our framework to another state-of-the-art modelling operator framework, illustrating that our approach predicts analytics outcomes accurately, and increases speedup. Furthermore, our vectorization model can project different real-world scenarios to a lower vector embedding representation accurately and distinguish them.

Keywords: Data Quality · Analytics Modelling · Vector embeddings · Vector Similarity

1 Introduction

Big data technologies daily face the rapid evolution in volume as well as variety and velocity of processed data [12]. Such big data characteristics routinely force analytics pipelines to underperform, requiring continuous maintenance and optimization. One major reason for this is bad data quality[1]. Poor data quality leads to low data utilisation efficiency and even brings forth serious decision-making errors [6].

Data quality can be improved when focusing on the actual content of the data. Data-centric Artificial Intelligence (AI) [30] emphasises on the quality, context, and structure of the data to improve its quality, as well as the analytical or machine learning (ML) algorithmic performance. Understanding the data context properties, such as data features, origins, relevance, and potential biases,

[1] https://tinyurl.com/de62sf48.

R. Wrembel et al. (Eds.): DEXA 2025, LNCS 16047, pp. 237–253, 2026.
https://doi.org/10.1007/978-3-032-02088-8_17

plays a critical role in modelling more accurate and reliable models. Data-centric AI prioritises the process of refining and enriching datasets to make them more suitable for real-world applications. Similarly, many researchers argue that prioritizing content-focused data quality is essential for achieving superior results [30].

Yet, the plethora of available data sources and datasets in an organisation data repository poses a significant challenge: Deciding the most suitable datasets for analytics workflows to ensure accurate results/predictions. While modern analytics workflows incorporate diverse operators, optimising dataset selection using data-centric AI methods remains an active research area [15]. When dataset selection is left to human experts, prediction performance drops, and it consumes more time. Equally costly and inefficient is the evaluation of all available datasets to identify high-quality inputs.

In previous work [9], predicting the output of an analytics operator assuming a plethora of available input datasets was tackled via the creation of an all-pair similarity matrix, which, relative to the similarity function used, reflected the distance between datasets over a single data quality metric (e.g., data distribution). Data or vector embeddings have been proposed to enhance big data analysis and modern AI systems. Data embedding vectorization [22,24] aims at projecting data from a high-dimensional representation space into a more compact, lower-dimensional space. Extracting meaningful information through data features using deep learning, data is projected to a lower representation space.

To improve the accuracy of a modelled analytic operator (i.e., predict the outcome of a ML algorithm without actually executing it due to its cost), we propose a framework that uses vector embeddings for dataset selection from a large data lake repository. Our method predicts an operator's output for an "unseen" query dataset, by selecting *qualitatively similar* datasets through similarity search over the vector embeddings. The selection of similar datasets reduces the prediction error, as well as the cost to model the operator, under the assumption that realistic analytical operators perform similarly under similar inputs. The embeddings are generated using our deep learning method, *NumTabData2Vec*, which processes entire tabular datasets rather than chunks or metadata, enabling efficient distinction between datasets and flexible modelling of multiple operators. Compared to similar previous work [4,9], our work uses state-of-the-art data representation (vector embeddings) which are able to capture multiple data properties that can be used in order to assess similarity, namely record order, dataset size, data distribution, etc.

The main contributions of our work can be summarised as follows:

- We introduce a framework for operator modelling (open-source prototype[2]) in order to predict its outcome on an unseen tabular input dataset from a plethora of available ones. Our method uses dataset vector embedding representations to improve the prediction performance via selecting the most relevant datasets to base its prediction upon.

[2] GitHub Repository.

- We develop a deep learning model architecture that transforms an entire tabular dataset of numerical values to a vector embedding representation.
- We provide an experimental evaluation of our proposed methodology using multiple real-world scenarios and compare it directly to the Apollo system [4,9].

Our evaluation illustrates that our methodology produces low prediction error by adaptively selecting similar quality datasets, achieving significant amortized speed-ups. *NumTabData2Vec* evaluation shows that it effectively projects datasets into vector embeddings while accurately capturing diverse dataset properties within the representation space.

2 Related Work

Prior efforts have focused on boosting algorithm performance by increasing data input (record number) rather than assessing quality. Consequently, we review works that identify optimal data features for analytic operator optimization. Vectorising data to lower embedding representation is a modern method that helps in identifying significant features across data types and datasets. As vector embeddings extract important features from data, we discuss studies that used the feature representation of data tuples to improve ML model prediction.

2.1 Data Quality

Big data applications aim to improve data quality by addressing various challenges. Dagger [26] enhances data quality by detecting pipeline errors using an SQL-like language, while ReClean [3] automates tabular data cleaning via reinforcement learning. IterClean [25] employs a large language model (LLM) to iteratively clean data by labelling initial tuples and using error detection, verification, and repair. In [8], data tuple quality is measured using Shapley values from game theory, with Truncated Monte Carlo Shapley and Gradient Shapley methods estimating a tuple's value to a learning algorithm. Apollo [4,9] is a content-based method predicts analytic operator outcomes by leveraging dataset similarity through three steps: creating a similarity matrix, projecting datasets to a lower-dimensional space, and modelling the operator using a small random subset of datasets. Unlike Apollo, our approach selects the most relevant, high-quality datasets to model analytic operators, aiming to improve prediction performance. Additionally, our vector embeddings incorporate all dataset properties, whereas Apollo's [9] similarity functions target only a single property.

2.2 Dataset Selection Inference

SOALA [11] selects optimal data features through online pairwise comparisons to maintain ML models over time. Its extension, Group-SOALA, introduces group maintenance to identify high-quality feature sets. In [23], the tf.data API framework enables the creation of ML pipelines focused on selecting relevant datasets and features to improve data quality. Similarly, our framework uses dataset vector embeddings to select the most suitable datasets for modelling analytic operators or ML models, enhancing prediction accuracy.

2.3 Data Vectorization and Embeddings

The goal of data vectorization is to project high-dimensional data into a lower-dimensional vector space. Word2Vec [22] (using Continuous Bag of Words and Skip-Gram) [22] leverages word context to generate embeddings. Graph2Vec [24] creates graph embeddings by dividing graphs into sub-graphs with a skip-gram model and aggregating their embeddings. ImageDataset2Vec [7] extracts meta-features from image datasets to generate embeddings, helping to select the most suitable classification algorithm. Dataset2Vec [16] uses meta-features and the DeepSet model to project datasets into embeddings and measure dataset similarity. Table 2Vec [31] generates table embeddings by incorporating data features, metadata, and structural elements like captions and column headings. Mix2Vec [32], is an unsupervised deep neural network that projects mixed data into vector embeddings. In a clustering experiment like in their work on the common Adult dataset, our model outperformed Mix2Vec (recent method without publicly available code) by nearly 10%, demonstrating superior performance. Inspired by these methods, we designed a model that generates vector representations of tabular datasets over their record data values, not their metadata.

Vector embeddings, which capture valuable information from data tuples, are widely used in classification tasks. TransTab [28] encodes features with transformer layers to predict classes, leveraging supervised and self-supervised pretraining. FT-Transformer [10] and Res-Net architectures similarly use embeddings of categorical and numerical features, processed through transformer layers for class prediction, while Tab-Transformer [14] combines embedded categorical and normalized continuous features in an MLP for class prediction. Unlike these tuple-level approaches, our framework uses dataset-level embeddings to identify relevant datasets, enhancing analytic operator performance.

3 Methodology

In this section, we describe our proposed framework for modelling analytic operators over a large number of available input datasets. We also describe our approach for vectorizing tabular datasets, *NumTabData2Vec*.

3.1 Framework Architecture

Consider a data lake repository that contains a (possibly large) number n of structured tabular datasets $D = (d_1, d_2, d_3, \dots, d_n)$. Also, let us consider an analytics operator (e.g., a ML algorithm) Φ and an "unseen" dataset D_o (from the same domain). We assume that each $D_i, 1 \leq i \leq n$ as well as D_o consist of records with numerical values only. Each dataset can, naturally, consist of

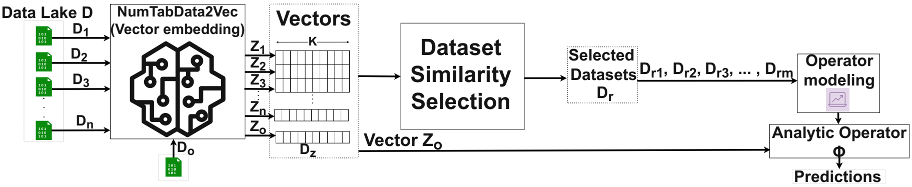


Fig. 1. Pipeline framework architecture

different number of records. Operator Φ consumes a single such dataset as input to produce a single numerical output: $\Phi : D_i \rightarrow \mathbb{R}$. Our goal is to predict $\Phi(D_o)$ with minimal cost and error by modelling the operator's output for D_o using a small subset of similar datasets $D_r \subseteq D$.

Datasets in D_r closely match D_o in their properties (e.g., order, distribution, and size to name a few). Previous work [4,9] had to use separate similarity functions for each such property. In contrast, we leverage the embedding vectorization (D_z) to efficiently identify the most similar datasets using all dataset properties.

Figure 1 depicts the pipeline of our proposed framework. Datasets in D are transformed into k-dimensional vectors using our *NumTabData2Vec* scheme, and these embeddings are stored for reuse. Each time a D_o needs to be inferred relative to an analytics operator Φ, its vector embedding is created. The datasets used for the creation of the model are selected via similarity search to produce a small subset of relevant datasets. These chosen datasets are then used to model and predict $\Phi(D_o)$, ensuring that only high-quality, pertinent data is processed. With this approach, our framework is utilizing "right quality" data in its inference mechanism, with irrelevant and extraneous datasets being excluded from the modelling process.

The datasets are embedded by our *NumTabData2Vec* method, which transforms each entire dataset in D into a k-dimensional vector z that captures all its characteristics. Our framework operates seamlessly across diverse real-world scenarios without modification, requiring only the specification of a repository containing distinct numerical tabular datasets. The Vector embedding z is a lower-dimension representation of the dataset with the entire characteristics of the dataset being encoded. Dataset D_o is similarly embedded as z_o. Using the embedding representation z_o and applying different similarity functions over the vector representations D_z, we may choose the most similar subset of D. The final step of the pipeline involves the operator modelling with any relevant method (e.g., Linear Regression, SVM, Multi-Layer Perceptron, etc.). This model is then used in order to infer the value of $\Phi(D_o)$.

3.2 NumTabData2Vec

This method transforms a dataset D_i into a lower-dimensional vector z using only its numerical values while excluding metadata like column names and file-

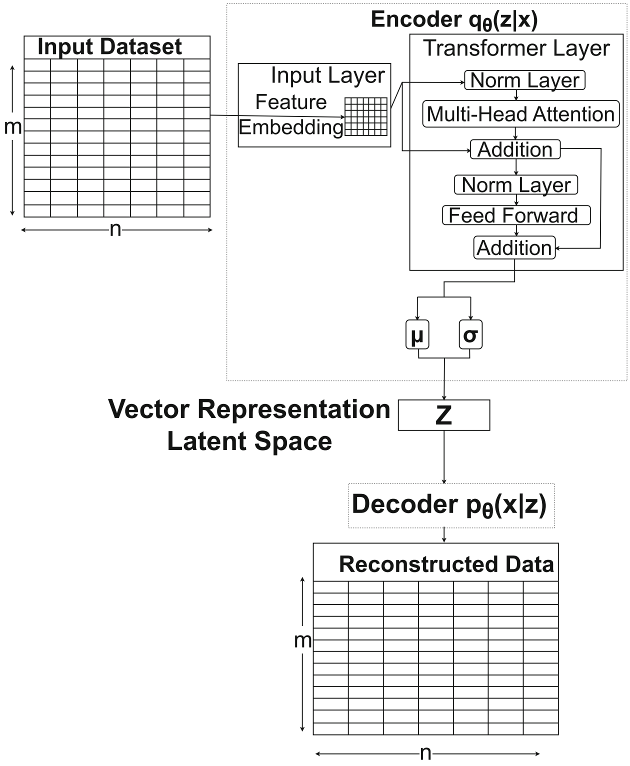


Fig. 2. NumTabData2Vec deep learning model architecture

names. We present a deep learning model architecture based on the variational autoencoder (VAE) [17] concept, that projects high-dimensional data into a $1 \times k$ vector embedding dimension. The proposed method is thus defined as:

$$NumTabData2Vec\,(D_i) \rightarrow z, \tag{1}$$

where the model takes an $m \times n$ dimensional numerical dataset and projects it to a lower k-dimension space z ($k > 1$). We desire our method to be generally applicable to any dataset by learning to project a vector embedding during training. We also expect this method to learn vector embeddings from diverse data and operate without additional training or fine-tuning. Finally, we ought our scheme to be able to quickly and precisely extract vectors from every input while handling varying dataset dimensions without modifications.

The deep learning model architecture is depicted in Fig. 2, where a dataset D_i with dimensions $m \times n$ passes through the encoder and is projected into a vector embedding representation z, then reconstructed by a decoder that mirrors the encode. The vector representation z is learned using a probabilistic encoder $q_\phi\,(z|x)$, and decoder $p_\theta\,(x|z)$ that learns the distribution utilising the Kullback-

Leibler (KL) divergence [18]. To achieve that, the following condition:

$$minD_{KL}\left(q_{\phi}\left(z|x\right)\|p_{\theta}\left(x|z\right)\right) \tag{2}$$

of Kullback-Leibler (KL) divergence must be minimised. To learn the new vector representation z, the dataset must be reconstructed back from z to its input format using the decoder, to verify that the vector representation is compact. This reconstruction loss is part of the overall loss function, defined as:

$$\mathcal{L}_{\theta,\phi}\left(x\right) = \mathbb{E}_{q_{\phi}(z|x)}\left(\log\left(p_{\theta}\left(x|z\right)\right)\right) - D_{KL}\left(q_{\phi}\left(z|x\right)\|p_{\theta}\left(x|z\right)\right) \tag{3}$$

This loss function is called evidences lower bound (ELBO). While the KL divergence is minimised to learn the vector embedding representation z, the ELBO is maximized so the condition,

$$argmax\mathcal{L}_{\theta,\phi}\left(x\right) \tag{4}$$

must be satisfied. The Decoder $p_{\theta}\left(x|z\right)$ is only used during the training phase to teach the encoder how to project the vector embedding representation z.

The decoder extracts feature embeddings from the input dataset D_i and processes them through Transformer layers. For the transformer layer we are using the pre-LN Transformer layer [29] instead of the traditional post-LN Transformer layer where the normalisation layer is employed inside the residual connection and before the prediction of the feed-forward layer. Following that, the transformed embedding space is projected into a probabilistic vector space z using the mean (μ) and standard deviation (σ). This lower-dimensional space retains all essential information about D_i, and a higher dimension k in z leads to a more accurate representation by capturing additional features [16].

3.3 Dataset Selection

The selection of the most similar datasets has been implemented using three different approaches. Different similarity functions are easily plugged into our pipeline. The first method uses cosine similarity, which measures the angle between two vectors in the embedding space z independent of their magnitudes, with a higher value indicating greater similarity. The alternative method calculates the Euclidean distance between two vectors to capture their geometric closeness and determine dataset similarity. This approach aids in selecting the most relevant datasets according to organizational requirements. The smaller the distance value then more similar are the datasets. Dataset selection using cosine similarity or euclidean distance selects a fraction of $\lambda, \lambda > 0$ of the closest datasets to dataset D_o. The third approach involves utilising the K-Means [19,21] clustering technique to choose the most relevant datasets. The datasets from D are divided into s ($s > 1$) separate clusters, where datasets with similar features are grouped together in the same cluster based on similarity equations. We determine the optimal number of clusters using silhouette scores [27]. This is done by the following equation:

$$SilhouetteScore(z_i) = \frac{b(z_i) - a(z_i)}{\max(a(z_i), b(z_i))}, \tag{5}$$

Algorithm 1. K-Means Clustering Algorithm for dataset selection

```
Require: Vectors $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$, $D_o$ vector $z_o$, range of clusters $S = (2, \ldots, p)$,
    Maximum size of cluster $max_s$, Minimum size of cluster $min_s$
Initialize the number of cluster $s$ using Silhouette score: $s = SilhouetteScore(Z, S)$
Initialize the $s$ cluster centroids $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_s\}$ randomly from the vectors $\mathbf{Z}$.
repeat
    Assignment Step:
    for each vector $\mathbf{z}_i \in \mathbf{Z}$ do
        Assign $\mathbf{z}_i$ to the nearest centroid based on Euclidean distance:
            Assign $\mathbf{z}_i$ to cluster $j = \arg\min_j \|\mathbf{z}_i - \mathbf{z}_j\|^2$
    end for
    Update Step:
    for each centroid $\mathbf{c}_j$ do
        Update $\mathbf{c}_j$ as the mean of all vectors assigned to cluster $j$:
            $\mathbf{c}_j = \frac{1}{|\{\mathbf{z}_i \in C_j\}|} \sum_{\mathbf{z}_i \in C_j} \mathbf{v}_i$
    end for
until Centroids $\mathbf{C}$ do not change significantly
Save the cluster model K-Means
Find in which cluster the vector $z$ belongs, $c = \textbf{K-Means}(z)$
Find which datasets $D_r$ are belongs to cluster $c$
Check the number of datasets in $D_r$ and update it if it does not meet the $min_s$ and $max_s$.
Return Datasets $D_r$
```

where for each vector z_i computes the mean intra-cluster distance $(a(z_i))$ which is the distance with the other vectors in the same cluster, and the mean nearest-cluster distance $(b(z_i))$ is the minimum average distance with the other vectors in a different cluster. Silhouette Score ranges from -1 to 1, and the higher value defines the best s number for clusters.

Algorithm 1, outlines the K-means process for selecting relevant datasets D_r based on the target dataset D_o. Using the optimal s (with the highest Silhouette score), the vector representations z of each dataset are clustered. Next, the algorithm uses the vector z_o of dataset D_o to find the closest cluster centroid. All datasets in that cluster are defined as the relevant datasets D_r, which are then used to model the analytics operator. However, we defined a maximum and minimum size for D_r, and if these conditions are not met, datasets are either removed from the cluster or added based on their distance from the cluster centroid. These small adjustments are only made in cases where the clustering technique does not yield results that satisfy our requirements.

4 Evaluation

We compared our framework with Apollo [4,9], which models analytic operators using data content. Two loss functions to measure prediction accuracy are employed: root-mean-square error (RMSE) and mean absolute error (MAE). RMSE is sensitive to outliers, while MAE is not; conversely, RMSE accounts for error direction, which MAE cannot. We further assess efficiency using *Speedup* and *Amortized Speedup* metrics, where *Speedup* is defined as $\frac{T_{op}^{(i)}}{T_{SimOp}^{(i)}+T_{vec}+T_{sim}+T_{pred}}$, where $T_{op}^{(i)}$ is the time to execute operator i on all datasets, $T_{SimOp}^{(i)}$ is the time to model the operator with datasets from similarity search, T_{vec} is the vector embedding computation time, T_{sim} is the similarity search time, and T_{pred} is the prediction time for D_o. Amortized speedup including one-time vectorization per data lake across multiple operators. Three variants with vector sizes 100, 200, and 300 (each with eight transformer layers) were trained for 100 epochs on four NVIDIA A10 GPUs. More experimental evaluation results can be found in the extended version of this work [20].

4.1 Evaluation Setup

Our framework is deployed over an AWS EC2 virtual machine server running with 48 vCPUs of AMD EPYC 7R32 processors at 2.40 GHz, and four A10s GPUs with 24 GB of memory each, 192 GB of RAM memory, and $2TB$ of storage, running over Ubuntu 24.4 LTS. Our code is written in Python (v.3.9.1) and PyTorch modules (v.2.4.0). Apollo was deployed on the same AWS EC2 virtual machine server, utilizing only the vCPUs and RAM, as it does not require a GPU for execution.

4.2 Datasets

We evaluated our framework using four real-world datasets (see Table 1). The NumTabData2Vec module was trained on separate data (60% training, 40% testing). The Household Power Consumption (HPC) dataset [13] contains 401 datasets with 2051 tuples and seven features recorded at one-minute intervals of electric power usage measurements. The Adult dataset [5], used for binary classification, predicts income levels and includes 100 datasets with 228

Table 1. Dataset properties for experimental evaluation

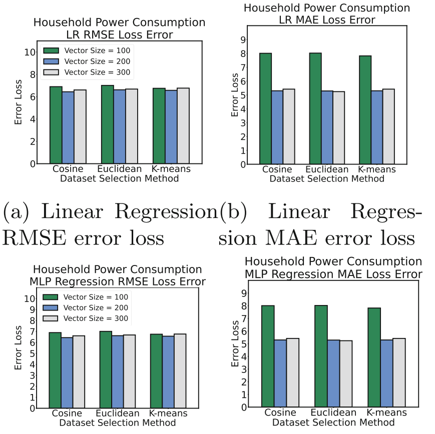
Dataset Name	# Files	# Tuples	# Columns
Household Power Consumption [13]	401	2051	7
Adult [5]	100	228	14
Stocks [1]	508	1959 – 13	7
Weather [2]	49	516	7

individuals and socio-economic features. The Stock Market dataset [1] consists of 508 datasets with 13 to 1959 tuples describing daily NASDAQ stock prices. Weather dataset [2] provides hourly measurements from 36 U.S. cities (2012–2017), split into 49 datasets with 516 tuples and seven features. Any categorical feature column in all datasets is transformed to numerical data by one-hot encoding. These datasets were selected to demonstrate our framework's ability to perform consistently across diverse real-world scenarios.

Our framework was evaluated by predicting the outputs of various ML operators without directly executing them. Datasets were projected into k-dimensional spaces with vector dimensions of 100, 200, and 300. For regression datasets (Household Power Consumption and Stock Market), we modelled Linear Regression (LR) and Multi-Layer Perceptron (MLP), while for classification datasets (Weather and Adult), we modelled Support Vector Machine (SVM) and MLP classifiers. Each experiment has executed 10 times, and we report the average error loss and speedup.

4.3 Evaluation Results

Figures 3, 4, 5, and 6 present the evaluation results of different similarity search methods across vector embedding spaces of sizes 100, 200, and 300 (green, blue, and grey bars, respectively). In each sub-figure, the y-axis represents the error loss value, while the x-axis displays the similarity search method applied over the vector embeddings. Figures 3 and 4 display results for the Stock Market and Household Power Consumption datasets, with MLP regression in the bottom sub-figure and LR in the top. Figures 5 and 6 show results for the Weather and Adult datasets, with SVM (SGD) in the top sub-figure and MLP classifier in the bottom. Left sub-figures use RMSE loss, while right sub-figures use MAE loss.



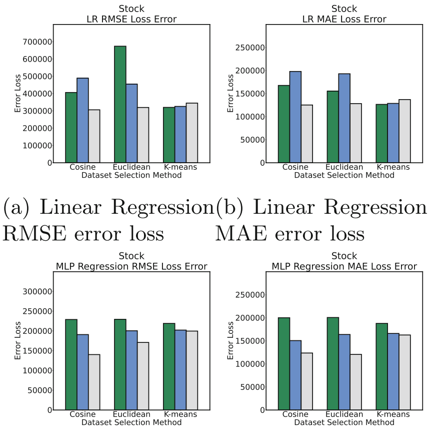
(a) Linear Regression RMSE error loss

(b) Linear Regression MAE error loss

(c) MLP for Regression RMSE error loss

(d) MLP for Regression MAE error loss

Fig. 3. Household power consumption dataset prediction error loss



(a) Linear Regression RMSE error loss

(b) Linear Regression MAE error loss

(c) MLP for Regression RMSE error loss

(d) MLP for Regression MAE error loss

Fig. 4. Stock market dataset prediction error loss

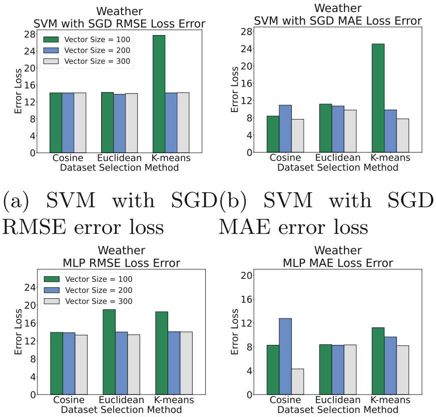
Table 2. Evaluation results of our framework exported analytic operator with lowest prediction error in comparison with Apollo

Dataset Name	Method	Operator	RMSE	MAE	Speedup	Amortized Speedup
Household Power Consumption	300V Cosine	LR	**6.61**	**5.42**	0.0017	**1.99**
	300V SR-0.2	LR	7.77	6.66	0.0018	1.42
	Apollo-SR 0.1	LR	2968.01	2352.55	**0.015**	0.024
	Apollo-SR 0.2	LR	2811.49	2229.50	0.015	0.024
	300V K-Means	MLP Regr.	**6.70**	**3.38**	0.9249	**1.99**
	Apollo-SR 0.1	MLP Regr.	3322.05	2606.99	2.38	1.74
	Apollo-SR 0.2	MLP Regr.	3850.01	2609.36	**2.38**	1.74
Stock	300V Cosine	LR	306382.28	125335.65	0.00085	**1.91**
	300V SR-0.4	LR	21861625.91	5674215.265	0.00087	0.33
	Apollo-SR 0.1	LR	**153665.92**	**118236.48**	**0.00093**	0.00096
	Apollo-SR 0.2	LR	166844.95	133306.68	0.00093	0.00096
	300V Cosine	MLP Regr.	**140236.47**	**123571.12**	0.63	**1.91**
	Apollo-SR 0.1	MLP Regr.	175150.82	145123.09	**0.93**	0.96
	Apollo-SR 0.2	MLP Regr.	174390.81	146338.73	0.93	0.96
Weather	300V Cosine	SVM SGD	**14.13**	**7.63**	1.06	**22.8**
	Apollo-SR 0.1	SVM	69.51	25.52	**2.10**	1.16
	Apollo-SR 0.2	SVM	68.70	22.81	2.10	1.16
	300V Cosine	MLP	**14.29**	**4.03**	1.03	**22.8**
	300V SR-0.4	MLP	15.95	13.31	1.02	1.77
	Apollo-SR 0.1	MLP	69.62	23.10	**1.34**	1.14
	Apollo-SR 0.2	MLP	673.56	**84.70**	1.32	1.14
Adult	300V Cosine	SVM SGD	**0.36**	**0.2**	0.37	**2.78**
	Apollo-SR 0.1	SVM	68.32	22.95	**0.75**	0.85
	Apollo-SR 0.2	SVM	68.88	22.88	0.74	0.85
	300V K-Means	MLP	**0.36**	**0.19**	0.30	2.78
	300V SR-0.2	MLP	6.01	6.00	0.54	**3.54**
	Apollo-SR 0.1	MLP	71.11	26.51	**1.07**	1.31
	Apollo-SR 0.2	MLP	70.16	25.74	1.05	1.31

Figure 3, for the HPC dataset, shows as increase the vector dimension size there is slightly lower prediction error for all the operator modelling. Different similarity methods do not result in any significant differences in the prediction error loss for all the operator modelling. This suggests that, regardless the similarity selection method, our framework effectively selects the most optimal subset of data to improve model predictions on the unseen input dataset D_o. Additionally, we observe higher error loss with a vector size of 100, which can be attributed to the reduced representation capacity of lower-dimensional vectors. This limitation results in fewer "right" datasets being selected.

For the stock market dataset, Fig. 4 depicts that a vector embedding representation of size 300 models more accurate operators, with cosine similarity performing best in the similarity search and modelling the most optimal operator. However, due to the inherent volatility in Stock market data from different days, all models in the stock market dataset experiments exhibit high loss values.

In the weather dataset, the SVM operator results (Figs. 5a and 5b) show that using 300 sized vectors in the representation space consistently led to more accurate operator models across all similarity methods. Specifically, cosine similarity in combination with the 300-dimensional vector embedding reduced the error rate in operator predictions, demonstrating that projecting datasets into this representation space and applying cosine similarity improves the prediction accuracy on the modelled operator. For the MLP classifier (Figs. 5c and 5d), the results illustrate that using vector embeddings of size 300 and Cosine similarity-produced the most accurate MLP classifier operators.



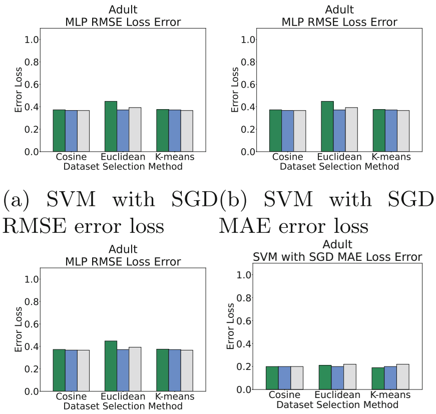
(a) SVM with SGD RMSE error loss

(b) SVM with SGD MAE error loss

(c) MLP RMSE error loss

(d) MLP MAE error loss

Fig. 5. Weather dataset prediction error loss



(a) SVM with SGD RMSE error loss

(b) SVM with SGD MAE error loss

(c) MLP RMSE error loss

(d) MLP MAE error loss

Fig. 6. Adult dataset prediction error loss

On the other hand, the Adult dataset shows the lowest error rates, with error loss values consistently below 0.5 across all vector embedding dimensions and similarity search methods (see Fig. 6). The Adult dataset, besides exhibiting a high number of rows, also has a higher number of columns, which demonstrates that our framework performs consistently well even with larger datasets. Additionally, we observe that the lowest prediction error across all datasets occurs when using higher-dimensional vector embeddings. With a trade-off between accuracy and execution time as the difference to generate all data lake available datasets vector embedding representation between 100 and 300 size dimension in the vector representation space to be less than 60 s. This confirms that a higher number of vector dimensions leads to more accurate predictions, consistent with findings in previous research [22].

We conducted an experimental evaluation using the Sampling Ratio (SR) approach, similar to Apollo [9], but employed neural networks built from the vector embeddings of each dataset. The SR approach involves a unified random selection of $l\%$ datasets from the vector representation space, using this subset to construct a neural network for predicting operator outputs. We tested SR values of 0.1, 0.2, and 0.4, as well as vector embedding dimensions of 100, 200, and 300, across all datasets.

For the HPC and Weather datasets, the SR approach was approximately 15% less accurate in operator prediction compared to all similarity search methods, even as vector embedding dimensions increased. In contrast, the Stock dataset exhibited a significantly larger discrepancy, with the SR approach performing about 70% worse in prediction accuracy across all vector embedding dimensions. Similarly, in the Adult dataset, the SR approach recorded the poorest performance, delivering nearly 90% worse prediction accuracy compared to the similarity search methods.

Table 2 compares model operators, loss functions, and speedup metrics for our framework and Apollo at SR values of 0.1 and 0.2. Methods 100V, 200V, and 300V denote vector embedding dimensions. The lowest prediction errors align with our pipeline's similarity search method. Apollo outperforms our framework on the Stock dataset for the LR analytic operator with the smallest amount of SR. However, our framework excels with the MLP regression operator, improving RMSE and MAE by 20% and 17%, respectively. The LR operator's performance gap on the Stock dataset is minor. For other datasets, our framework consistently surpasses Apollo across different SR values. This demonstrates the effectiveness of our similarity search approach, which enhances data quality and reduces Φ prediction errors by identifying relevant datasets D_r from the data lake directory D. The Adult dataset also highlights our framework's advantage with increasing feature dimensions. Although Apollo achieves better raw speedup due to the higher complexity of our framework's vectorization step, our framework outperforms it in amortized speedup. By excluding the reusable vectorization process, it achieves speed gains of 10% to 60% for most operators. The SR approach, leveraging vector embedding representations, enhances operator prediction compared to Apollo and achieves greater amortized speedup. However, the similarity search method outperforms both Apollo and the SR approach in prediction accuracy and amortized speedup, establishing its clear superiority across most datasets and operator scenarios.

4.4 NumTabData2Vec Evaluation Results

Table 3. Similarity between vectors of different datasets scenarios

NumTabData2Vec Vector Size	Similarity
100	0.54
200	0.18
300	0.16

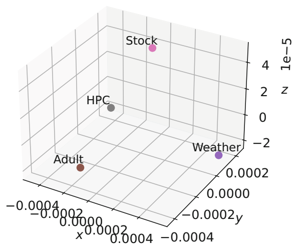


Fig. 7. Vector representation for each dataset from NumTabData2Vec

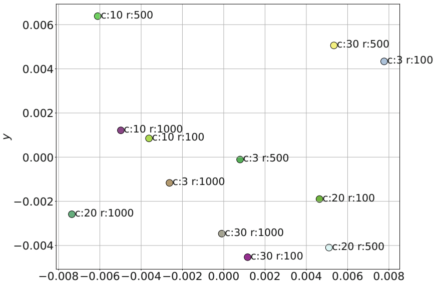


Fig. 8. Synthetic data vector embedding representation

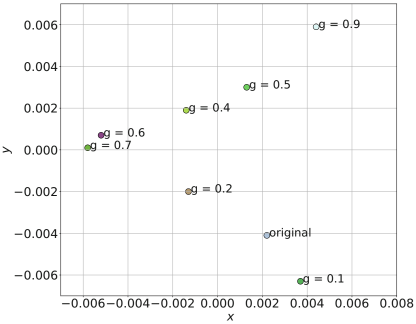


Fig. 9. HPC Dataset vector embedding representation with addition of Noise

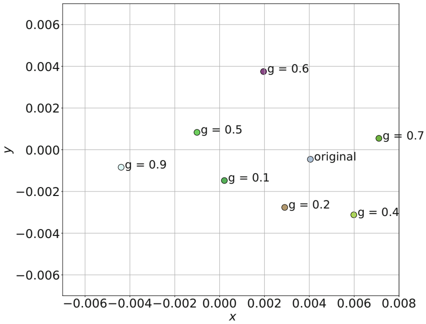


Fig. 10. HPC Dataset vector embedding representation with addition of Noise in the first column

Our proposed model, *NumTabData2Vec*, was evaluated for its ability to distinguish dataset scenarios based on qualitative differences. For each scenario, n random datasets were reduced to 3D vector embeddings using PCA, as shown in Fig. 7, which demonstrates *NumTabData2Vec*'s ability to distinguish datasets with minimal overlap across contexts. Unlike prior methods [22,24], which focus on text or graphs, *NumTabData2Vec* applies to entire datasets. Table 3 further highlights the average cosine similarity between dataset embeddings, showing greater dissimilarity as vector dimensions increase. However, results suggest that dimensions between 100 and 300 are sufficient for accurate distinction, avoiding the need for larger vector sizes.

To evaluate *NumTabData2Vec*'s ability to distinguish datasets with varying row and column counts, we generated synthetic numerical tabular datasets of different dimensions and vectorized them. Figure 8 shows datasets with columns ranging from three to thirty and rows from ten to one thousand, projected from a 200-dimensional space to 2D using PCA. Each bullet caption c and r corresponds to the columns and rows of the dataset, respectively. Datasets with the same number of columns cluster closely in the representation space, and a similar pattern is observed for datasets with the same number of rows. These results indicate that our method effectively distinguishes datasets based on size during vectorization.

To evaluate *NumTabData2Vec*'s ability to distinguish datasets by distribution and order, Gaussian noise was added to $l\%$ of tuples in an HPC dataset. Figure 9 shows the original and noise-modified datasets projected to 2D using PCA, with greater noise causing larger shifts in the representation space. This demonstrates the model's effectiveness in capturing distribution differences and distinguishing datasets based on ordering.

To assess fine-grained distinctions, we repeated the experiment by adding Gaussian noise exclusively to the first column of the dataset. Figure 10 shows the 2D vector space, where g in the bullet caption indicates the noise level. As noise increases, the representation shifts further from the original dataset, though it remains closer than in Fig. 9, with points more tightly grouped in 2D space.

5 Limitations

There exist a number of limitations in our work as we described it. In this section we briefly highlight them. Firstly, our input datasets comprise records of specific size and type (numerical). This currently excludes data with textual and categorical attributes, or tables with varying number of features inside a set of datasets. Secondly, we currently consider single-input and single-output operator modelling. Finally, our proposed NumTabData2Vec model for data vectorization has a performance limitation, as it cannot deal with datasets bigger than about 3000 tuples. This is mostly a hardware limitation of off-the-shelf GPUs (with at most 24GB of memory available for a budget GPU).

6 Conclusion

In this paper, we presented a novel framework for the modelling of an analytic operator (such as a ML algorithm) when a large number of input data is available and thus no brute-force execution can be performed. We propose a deep learning model, *NumTabData2Vec*, which transforms a dataset to a lower k-dimensional representation space z. Our framework produces vector embeddings for the input datasets using *NumTabData2Vec* and performs a similarity search to identify the most relevant subset of datasets for any unseen input. By modelling the analytic operator based on this selected subset, we are able to accurately predict its output on any given input dataset. In practice, we demonstrated that our framework can accurately model various common algorithms and compared favourably against a similar recent framework [9], in both accuracy and speedup. Furthermore, we showed that *NumTabData2Vec* can create different vector representations for datasets from different scenarios. We also demonstrated that *NumTabData2Vec* can effectively detect when noise is introduced into a dataset.

Acknowledgments. This work is partially supported by the project RELAX-DN, funded by the European Union under Horizon Europe 2021–2027 Framework Programme Grant Agreement number 101072456.

References

1. Stock Market Dataset. Kaggle (2020). https://www.kaggle.com/datasets/jacksoncrow/stock-market-dataset/data
2. Weather Dataset. Kaggle (2020). https://www.kaggle.com/datasets/selfishgene/historical-hourly-weather-data
3. Abdelaal, M., Yayak, A.B., Klede, K., Schöning, H.: Reclean: reinforcement learning for automated data cleaning in ml pipelines. In: 2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW), pp. 324–330. IEEE (2024)
4. Bakogiannis, T., Giannakopoulos, I., Tsoumakos, D., Koziris, N.: Apollo: a dataset profiling and operator modeling system. In: Proceedings of the 2019 International Conference on Management of Data, pp. 1869–1872 (2019)
5. Becker, B., Kohavi, R.: Adult. UCI Machine Learning Repository (1996). https://doi.org/10.24432/C5XW20
6. Cai, L., Zhu, Y.: The challenges of data quality and data quality assessment in the big data era. Data Sci. J. **14**, 2–2 (2015)
7. Dias, L.V., Miranda, P.B., Nascimento, A.C., Cordeiro, F.R., Mello, R.F., Prudêncio, R.B.: Imagedataset2vec: an image dataset embedding for algorithm selection. Expert Syst. Appl. **180**, 115053 (2021)
8. Ghorbani, A., Zou, J.: Data shapley: equitable valuation of data for machine learning. In: International Conference on Machine Learning, pp. 2242–2251. PMLR (2019)
9. Giannakopoulos, I., Tsoumakos, D., Koziris, N.: A content-based approach for modeling analytics operators. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 227–236 (2018)
10. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. Adv. Neural. Inf. Process. Syst. **34**, 18932–18943 (2021)
11. Gupta, N., et al.: Data quality toolkit: automatic assessment of data quality and remediation for machine learning datasets. arXiv preprint arXiv:2108.05935 (2021)
12. Hazen, B.T., Boone, C.A., Ezell, J.D., Jones-Farmer, L.A.: Data quality for data science, predictive analytics, and big data in supply chain management: an introduction to the problem and suggestions for research and applications. Int. J. Prod. Econ. **154**, 72–80 (2014)
13. Hebrail, G., Berard, A.: Individual Household Electric Power Consumption. UCI Machine Learning Repository (2006). https://doi.org/10.24432/C58K54
14. Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z.: Tabtransformer: tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020)
15. Jakubik, J., Vössing, M., Kühl, N., Walk, J., Satzger, G.: Data-centric artificial intelligence. Bus. Inf. Syst. Eng. 1–9 (2024)
16. Jomaa, H.S., Schmidt-Thieme, L., Grabocka, J.: Dataset2vec: learning dataset meta-features. Data Min. Knowl. Disc. **35**(3), 964–985 (2021)
17. Kingma, D.P., Welling, M., et al.: An introduction to variational autoencoders. Found. Trends® Mach. Learn. **12**(4), 307–392 (2019)
18. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)

19. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
20. Loizou, A., Tsoumakos, D.: Analytics modelling over multiple datasets using vector embeddings. arXiv preprint arXiv:2502.17060 (2025)
21. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, pp. 281–297 (1967)
22. Mikolov, T.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 **3781** (2013)
23. Murray, D.G., Simsa, J., Klimovic, A., Indyk, I.: tf. data: a machine learning data processing framework. arXiv preprint arXiv:2101.12127 (2021)
24. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: learning distributed representations of graphs. arXiv preprint arXiv:1707.05005 (2017)
25. Ni, W., Zhang, K., Miao, X., Zhao, X., Wu, Y., Yin, J.: Iterclean: an iterative data cleaning framework with large language models. In: Proceedings of the ACM Turing Award Celebration Conference-China 2024, pp. 100–105 (2024)
26. Rezig, E.K., et al.: Dagger: a data (not code) debugger. In: CIDR 2020, 10th Conference on Innovative Data Systems Research, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings (2020)
27. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
28. Wang, Z., Sun, J.: Transtab: learning transferable tabular transformers across tables. Adv. Neural. Inf. Process. Syst. **35**, 2902–2915 (2022)
29. Xiong, R., et al.: On layer normalization in the transformer architecture. In: International Conference on Machine Learning, pp. 10524–10533. PMLR (2020)
30. Zha, D., et al.: Data-centric artificial intelligence: a survey. ACM Comput. Surv. **57**(5), 1–42 (2025)
31. Zhang, L., Zhang, S., Balog, K.: Table2vec: neural word and entity embeddings for table population and retrieval. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1029–1032 (2019)
32. Zhu, C., Zhang, Q., Cao, L., Abrahamyan, A.: Mix2vec: unsupervised mixed data representation. In: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), pp. 118–127 (2020). https://doi.org/10.1109/DSAA49011.2020.00024