I.J. Information Technology and Computer Science, **2025, 5, 84-99**
Published Online on October 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2025.05.07

Data Optimization through Compression Methods Using Information Technology

Igor V. Malyk*
Department of Mathematical Problems of Control and Cybernetics, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58000, Ukraine
E-mail: i.malyk@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-1291-9167
*Corresponding author

Yevhen Kyrychenko
Department of Mathematical Problems of Control and Cybernetics, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58000, Ukraine
E-mail: kyrychenko.yevhen@chnu.edu.ua
ORCID iD: https://orcid.org/0009-0005-6150-5410

Mykola Gorbatenko
Department of Mathematical Modeling, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58000, Ukraine
E-mail: m.gorbatenko@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-1702-8785

Taras Lukashiv
Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Belvaux, L-4370, Luxembourg
E-mail: taras.lukashiv@uni.lu
ORCID iD: https://orcid.org/0000-0002-1651-6402

Received: 25 May 2025; Revised: 20 July 2025; Accepted: 26 August 2025; Published: 08 October 2025

Abstract: Efficient comparison of heterogeneous tabular datasets is difficult when sources are unknown or weakly documented. We address this problem by introducing a unified, type-aware framework that builds *compact data representations (CDRs)*—concise summaries sufficient for downstream analysis—and a corresponding *similarity graph* (and tree) over a data corpus. Our novelty is threefold: (i) a principled vocabulary and procedure for constructing CDRs per variable type (factor, time, numeric, string), (ii) a weighted, type-specific similarity metric we call *Data Information Structural Similarity (DISS)* that aggregates distances across heterogeneous summaries, and (iii) an end-to-end, cloud-scalable realization that supports large corpora. Methodologically, factor variables are summarized by frequency tables; time variables by fixed-bin histograms; numeric variables by moment vectors (up to the fourth order); and string variables by TF–IDF vectors. Pairwise similarities use Hellinger, Wasserstein (p=1), total variation, and L1/L2 distances, with MAE/MAPE for numeric summaries; the DISS score combines these via learned or user-set weights to form an adjacency graph whose minimum-spanning tree yields a similarity tree. In experiments on multi-source CSVs, the approach enables accurate retrieval of closest datasets and robust corpus-level structuring while reducing storage and I/O. This contributes a reproducible pathway from raw tables to a similarity tree, clarifying terminology and providing algorithms that practitioners can deploy at scale.

Index Terms: Information Technology, Data Similarity, Compressed Copy of Tabular Data, Compact Data Representation.

1. Introduction

Comparing tabular datasets collected from unknown or weakly documented sources is a recurring challenge in data engineering and machine learning. Full copies of data are often costly to move, store, and scan, while schema drift and heterogeneous variable types hinder direct, record-level comparison. What practitioners need is an interpretable, type-aware way to summarize each dataset, measure how similar two datasets are, and organize an entire corpus by similarity—without accessing all raw records. Prior work addresses related aspects for specific modalities (e.g., images, graphs) or assumes known schemas and model-centric embeddings [6, 16, 11, 13]. However, a unified approach that (i) produces compact, interpretable summaries per variable type, (ii) defines principled distances across types, and (iii) scales to large corpora in production remains under-specified.

Goal and scope. We study the problem of corpus-level comparison of tabular data under minimal assumptions on provenance and schema. Our objective is to construct a *similarity graph* (and associated tree) over datasets using only compact, task-oriented summaries rather than full data copies. We use the term *compact data representation (CDR)* to denote such summaries.

Novelty and Contributions

This paper makes the following contributions:

- Type-aware compact representations (CDRs). We formalize a vocabulary and procedure for building CDRs per variable type: frequency tables for factor variables, fixed-bin histograms for time variables, moment vectors (up to 4th order) for numeric variables, and TF–IDF vectors for string variables [25]. The design choice prioritizes interpretability and portability across tools.
- A unified similarity metric. We introduce *Data Information Structural Similarity (DISS)*, a weighted aggregation of type-specific distances—Hellinger, Wasserstein-1, total variation, L1/L2 norms, and MAE/MAPE—chosen to reflect distributional differences rather than single-point errors [26]. DISS yields a single score per dataset pair and an adjacency graph suitable for downstream algorithms.
- From graph to tree. We construct a corpus-level *similarity graph* whose minimum-spanning tree produces an interpretable *similarity tree*, enabling nearest-neighbor search and hierarchical exploration without assuming a fixed schema.
- Cloud-scale realization. We provide an end-to-end realization on AWS using Apache Spark on EMR for distributed summarization, S3 for storage, Airflow for orchestration, Glue for cataloging, and Athena for ad-hoc querying [22, 24, 23, 28, 29, 27]. The implementation targets large, multi-source CSV corpora and emphasizes reproducibility and operational transparency.

Methodology Overview

Our pipeline proceeds in four stages:

- Variable typing. Columns are classified as factor, time, numeric (continuous/discrete), or string using heuristics on uniqueness ratios, parseability, and structural cues (e.g., fixed formats for identifiers and dates).
- CDR construction. For each typed variable, we emit a compact, interpretable summary: (i) factor → value frequencies; (ii) time → fixed-bin histogram with shared bin edges for comparability; (iii) numeric → moment vector $(N, \sum x, \sum x^2, ..., \sum x^4)$; (iv) string → TF–IDF over a bounded n-gram vocabulary
- Pairwise similarity (DISS). We compute type-specific distances—Hellinger/TV for categorical distributions, Wasserstein- 1 for histograms, MAE/MAPE or Euclidean for moment vectors, and L1/L2 for TF–IDF—then aggregate them via
- a weight vector to obtain DISS. Weights can be user-set or learned on labeled pairs, depending on the application.
- Graph and tree construction. We build an adjacency graph from DISS scores and derive a minimum-spanning tree to obtain a similarity tree that supports corpus navigation, nearest-neighbor retrieval, and cluster discovery.

Practical Use and Deployment

The proposed approach is designed for production settings where data volume, heterogeneity, and governance constraints matter:

- Data discovery and cataloging. CDRs serve as lightweight, queryable descriptors for rapid dataset triage and semantic search across data lakes.
- Quality monitoring and drift detection. Periodic recomputation of CDRs highlights distributional shifts in numeric, categorical, temporal, or text fields before model retraining.
- Duplicate/near-duplicate detection. DISS identifies closest datasets, aiding de-duplication and lineage checks in multi-source ingestion.
- Privacy-aware comparison. Because CDRs summarize distributions rather than raw records, cross-team or cross-account comparisons can be performed with reduced exposure of sensitive data characteristics.
- MLOps and dataset selection. Nearest-neighbor retrieval on the similarity graph accelerates selection of training data that best matches target domains.

Operationally, the system reads raw CSVs from S3, constructs CDRs via PySpark on EMR, stores results in Parquet, registers schemas in Glue, and exposes them to Athena for analysis; Airflow orchestrates the end-to-end DAG with logging and retries [22, 27, 28, 29, 23]. This design reduces I/O and storage, keeps summaries interpretable, and scales to large corpora while remaining compatible with standard cloud tooling. In the remainder, we detail variable typing and CDR design, formalize the DISS metric, and describe graph construction, followed by the cloud implementation and case studies. A literature review situates our approach among representation, compression, and similarity methods for heterogeneous data.

1.1. Motivation

Let us highlight several main subtasks for which determining the similarity between data and storing compact copies of data plays a key role:

- Data-Driven. Data-driven techniques are used to analyze changes in the data's main characteristics when deciding to recalculate a particular model. Typically, such recalculations are carried out to test the central hypothesis that there is no significant change in the data when constructing mathematical data models based on machine learning algorithms. In [1], primary attention is paid to the study of model changes based on changes in data hyperparam- eters, based on which the data is modelled. The article shows some continuity between data hyperparameters and the corresponding models built based on the data. In addition, some aspects of data changes and metrics between the data are considered.
- Cloud computing and data transfer. As noted in [2], data transfer between different devices can lead to system overload and requires the creation of compact copies of data, which would allow the conclusion of the data without analyzing the data on each device separately, given the fact that the models will most often be the same. Thus, the transfer of compact data copies significantly simplifies the data processing and transmission procedure.
- Construction machine learning models and their analysis [3]. When analyzing data in machine learning mod- els, only some data characteristics are used without considering all other data properties. Let us give one simple motivational example, which will help us understand this point's essence in more detail. For this, let us assume that the data is stored in a tabular format, as shown in Table 1, and also think that some column of the data describes the dependent variable y. The other columns describe a set of dependent variables (x1, . . . , xp). We will also assume that the primary model, which is considered based on multiple linear regression with the least squares method as the primary method of estimating the model parameters.

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

In this case, the parameter estimates can be found as follows

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{1}$$

where

$$X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Np} \end{pmatrix}, y = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{pmatrix}$$

Thus, to reproduce the model, it is not necessary to consider the entire data set; it is only required to use some data characteristics. When estimating (1), these characteristics are numerical characteristics of the data, namely the mean values and the covariance matrix of the data.

The motivational example discussed above suggests that the compact copy of the data will directly depend on the purpose of the copy, or instead on the machine learning model for which the copy is created or on the hypothesis that should be tested using the compact copy. Using some techniques of machine learning algorithms, we will highlight several main criteria for the compact copies of the data we have considered, which we will take into account when construction:

- Size. The size of the created data copies plays a key role since this indicator plays a key role in creation. Usually, the copy size should be several times smaller than the original, but it will also depend on the speed of the system used to generate this copy. Therefore, we impose the following condition on the size of the data copy

$$Size_{CD} \ll Size_B,$$

where $Size_{CD}$ is the size of the compact data copy (in bytes), and $Size_B$ is the threshold value of the data size that will burst when creating a copy.

- Model. A compact copy of the data should contain the characteristics to create the primary model, based on which conclusions will be drawn about the relationship in the data, and it is possible to conclude the hypotheses put forward in the studies. As noted above, for a linear model, such characteristics are numerical characteristics that fully allow you to reproduce the regression and estimate the regression coefficients. It should be noted that most models will enable you to use only some numerical data characteristics, ignoring all others. It should be noted here that in the case of using ensemble machine learning methods, the indicators will be duplicated depending on the size of the ensemble method [3], so again, it will be necessary to analyze the size of the ensemble method and the characteristics based on which each model is calculated.
- Main research hypotheses. Testing the study's main hypotheses and answering the research questions require various statistical hypotheses, which are again based on numerous indicators of tabular data. This point is closely intertwined with the second point since statistical tests, similar to the main machine learning models, require nu- merical or other characteristics of the data, taking into account the specifics of the primary model.
- Relationship with other data and possibility of model expansion. This item includes the ability to save some characteristics not directly calculated from the data but determined from other sources. This can be, for example, the resource from which the data is taken, previous models that were used to analyze earlier versions of the data in data-driven, data preprocessing procedures, etc.

1.2. Literature Review

As mentioned above, the issue of creating compact copies of data and standardizing data sets is the subject of a large number of works devoted to both different areas of research and other assumptions about the type of data. Firstly, it should be noted that the main approaches are divided depending on the study of structured and unstructured data [4]. Since the purpose of this work is to consider the tabular data presented in Table 1, we will pay more attention to structured data analysis. As mentioned above, the primary focus is on tabular data, where each column represents a particular variable. Since we are interested in machine learning models, one of the main characteristics when creating a compact copy of data will be the type of column in the data. Based on the requirements for the vast majority of machine learning models, we will highlight the following types of columns, which we will consider below:

- Numerical variables. Variables that represent numerical characteristics of some object or phenomenon are the easiest to process and are used most often in calculations. Conventionally, these variables can be divided into subcategories depending on the characteristics being studied, such as integer variables, fractional variables, etc.
- Factor variables. This variable type characterizes quantities that can take on multiple values, usually not numerical, but only characterize levels. This type of variable includes binary, nominal, and ordinal variables. It should be noted that many machine learning models, including linear regression, require the conversion of factor variables into dummy numeric variables.
- Time variables. Variables that represent date and time in a predefined format.
- String variables. This variable type represents information in string form, without a constant length, a clearly defined structure, and specific fixed rules for creation. This variable most often characterizes the results of communication in social networks and is used to determine respondents' general moods, voting conditions, and emotional state [5].

First of all, let us consider examples of compression of graphic data (drawings, graphs, etc.). A representative of this type of work is the work [6], which proposes a reduction in the dimensionality of data using Bag-of-Features (BOF). In this work, the representation of drawings is proposed based on the density of the distribution of pixel colours in RGB representation based on some family of distributions $M = \{p(*\,|\,\theta), \theta \in \Theta \subset R^s\}$, where θ *i*s the parameter of the distribution of the family with densities $p(*|\theta)$, and the authors propose the use of the Riemann metric

$$D_F(\theta_1,\theta_2) = \min_{\substack{\theta(*):\\ \theta(0)=\theta \\ \theta(1)=\theta_2}} \int_0^1 \sqrt{\left(\frac{d\theta}{dt}\right)^T I(\theta) \left(\frac{d\theta}{dt}\right)} dt \tag{2}$$

where $I(\theta)$ is the Fisher information, the elements of which are determined from the ratio

The authors of this work compared the distance based on Fisher's information (2). They indicated several advantages in reducing the dimensionality of the data compared to other metrics, namely the cosine distance, Euclidean distance, Helling distance, and Gaussian distance (Gaussian kernel). A more detailed analysis of drawings of geometric objects is considered in the work [7], in which geometric objects are proposed to be represented as compact topological objects (half-edge data structures). From a geometric point of view, the corresponding representation is based on the use of a *k*-dimensional manifold *M* (or related *k*-dimensional manifolds) with the indication of appropriate restrictions on these manifolds:

- (k-1)-measurable cell complex of the manifold M is identical to one or two k- dimensional cells of the manifold M;
- the open star of a vertex in the manifold M is homeomorphic to an open subset of either R^k or R_+^k .

The authors show that the representation of geometric shapes based on k-dimensional manifolds allows, on the one hand, the construction of a compact version of the object and, on the other hand, the preservation of the object's detail. Using this approach, the authors achieved high accuracy for face recognition, and the recognition accuracy increased by 5%-10% compared to classical methods. The Bidirectional Reflectance Distribution Function (BRDF) was used to create compressed copies of geometric objects [8]. The work is based on the creation of a mathematical model of data representation based on distribution densities, and one of the main assumptions of the work is the assumption of normality of the density distribution of the object image. This assumption allows us to significantly simplify theoretical calculations and reduce the generalized parametric families considered in [6] to two-parameter families $N(a, \Sigma)$. Another approach to data classification is the construction of latent variables [9], which can be treated as generalized variables in the principal component method. This paper presents the CLaRe framework, which analyses three datasets. It is shown that the developed approach to constructing latent variables gives better classification and clustering results than classical machine learning methods. A compact representation of large-scale spatial data is considered in [10], in which the representation of spatial data is based on the use of the minimum bounding rectangle (MBR). The concept of MBR is introduced due to its ease of use, namely, for a spatial two-dimensional region a, the MBR is determined by the following relation.

$$MBR(a) = [x^-(a), x^+(a)] \times [y^-(a), y^+(a)] \tag{3}$$

where

$$x^-(a) = \inf_x \{(x,y) \in a\},\ x^+(a) = \sup_x \{(x,y) \in a\}$$

$$y^-(a) = \inf_y \{(x,y) \in a\},\ y^+(a) = \sup_y \{(x,y) \in a\}$$

Thus, the representation of spatial data in R^2 is proposed using coverage of rectangles and their intersections. Compared to representing spatial data in a polygon, this approach significantly reduces the required data to define a spatial object. The authors of this work present a method for optimal representation of the spatial domain a R^2 or a R^3, i.e., minimizing the rectangular MBR(a) in definition (3).

A more formal approach to the classification of tabular data is considered in [11]. Based on this approach, tabular data are represented as a triple (F, X, y), where F is the column specifications, X is a set of independent variables, y is a dependent variable that must be evaluated in the regression or classification problem. Secondly, a new neural network architecture is developed for regression and comparison of tabular data, and it is shown that the accuracy of the developed neural network exceeds the number of classical methods (XGBoost, CatBoost, gcForest, Net-DNF and FCNN) and the speed of the proposed neural network is not less than in classical methods. The main disadvantage of the developed method is that the authors assume that the specification F is known, although in most cases, this assumption is very optimistic when comparing data. A slightly different approach to the representation of tabular data based on the definition of their complexity is presented in [12], which proposes to use the complexity of questions according to the Wood formalization:

$$QC_i = |IC_i| + |R_i| + |A_i|$$

where IC_i is the number of information clues presented in the question i, R_i is the number of relationships that the participant must identify, and A_i is the number of judgments that must be made regarding the set of elements. Thus, the approaches considered by the authors allow for the creation of encrypted copies of data (surveys) with a high degree of protection. Another interesting question raised in this work is comparing tabular and graphical data from the point of view of reliability and cryptoresistance. The authors show that the tabular representation is more reliable and has several advantages over the graphical representation. Another approach to creating compressive copies of data based on neural networks is considered in work [13], which proposes using a neural network to create a compressed copy of data. This approach is close to the work approach [11], and in both cases, convolutional neural networks are proposed. Thus, compressed data in works [11, 13] are considered data obtained at the intermediate steps of a neural network. As noted above, for analysing machine learning models, only a particular set of numerical characteristics of a specific data set is usually required, which may include a vector of means, a covariance matrix, tensors of the third and fourth moments, etc. Theoretical aspects of the study of data space reduction and, accordingly, the creation of compressed copies of data are presented in the work [14], in which the authors consider the possibility of reducing the dimensionality of the original data from n to $\frac{C\,In(n)}{In(In(n))}$ for sufficiently large n, where C is some universal constant. Thus, the authors of this work indicate the possibility of reducing the dimensionality of data based on the calculation of the necessary moment functions with a predetermined accuracy. Another approach to using neural networks to construct compressed copies of data is considered in [15], which, unlike works [11, 13], proposes a step-by-step data convolution using convolutions, i.e., new factors are constructed at each neural network layer

$$\tilde{x}_i^k = d(\tilde{x}_i^{k-1}, \tilde{x}_i^{k-2})$$

where $\tilde{x}_i^k$ is the value of variable x_i at k-th iteration. The authors of the work show that in classification problems, the developed approach allows the preservation of all the relationships in the initial data, filtering out noise and increasing the accuracy of the estimation in comparison with classical methods (Random Forest, XGboost, LightGBM, CatBoost, TabNet, ReConTab).

It is also necessary to mention the creation of compressive copies of graphs as mathematical moles of many real phenomena. It should be noted that data compression in the case of graph-based representation has several features related primarily to the possibility of reducing the graph by removing vertices or edges. It should be noted that some authors suggest always considering complete graphs since removing certain actors (graph vertices) or relationships (graph edges) can lead to a complete change in the group structure for which this graph is constructed as a mathematical model. The authors of [16] consider graphs as database models for storing and comparing tabular data. The tabular data or its attributes are considered as vertices, and as edges, various types of relationships between these vertices are considered. The authors considered and compared the following systems: DEX, Neo4j, or HyperGraph, while others, such as SAP, HANA Graph and SQLGraph, and each of the systems has both its advantages and disadvantages. Based on the methodology proposed by the authors, the data should be represented in the form of a ten-dimensional tuple

$$G = (L_N, L_E, N, E, R, L_A, S_N, S_E, A_N, A_E) \tag{4}$$

where

- L_N is the set of possible labels that can be taken by the nodes of the graph;
- L_E is the set of possible labels that can be taken by the edges of the graph;
- $N = (n_i, l_j)$ is the set of vertices of the graph;
- $E = (e_i, l_j)$ is the set of edges of the graph;
- R contains the links between the nodes, i.e. the initial and target nodes of the edges;
- $L_A = \{a_i\}$ is the set of different attribute labels of the graph. In other words, L_A is the set of all different attributes that describe the nodes and edges of the graph
- $S_N = \{sn_i\}$ is the set of schemes for node types;
- $S_E = \{se_i\}$ is the set of schemes for edge types, defined in a completely similar way to S_N. Each element of S_E defines a valid scheme for an edge type;
- $A_N = \{n_i, a_j, \upsilon_k\}$ defines the properties of the nodes;
- $A_E = \{e_i, a_j, \upsilon_k\}$ describes the properties.

As we can see, the representation (4) is quite complete and contains all the necessary data for the construction of a data schema. However, this approach will work well only if the properties of the tabular data, i.e. the data structure itself, are known.

Another direction in the context of the study of the work is the method of data reconstruction based on the construction of variables with specified associations [17, 18]. In the work [17], as noted above in the introduction, it is proposed to use, along with the numerical characteristics of the variables, their covariance or correlation matrices, i.e. each variable will be characterized as a set of moments

$$\upsilon ar_i : \left\{ m_i^1 = \frac{1}{N}\sum_{i=1}^{N} x_{ij}, \ldots, m_i^k = \frac{1}{N}\sum_{i=1}^{N} x_{ij}^k \right\} \tag{5}$$

and a correlation matrix or correlation matrices between different sets of variables (meaning numerical variables and factor variables):

$$R = \left(r_{ij} = \frac{1}{N\sqrt{m_i^2 m_j^2}} \sum_{l=1}^{N} (x_{il} - m_i^1)(x_{jl} - m_j^1) \right) \tag{6}$$

where k is the maximum order of the moment function in the representation (5). In [18], the problem of creating new classes when creating labels in clustering and semi-clustering problems is also partially considered. This problem closely overlaps with the topic of our study since, as will be shown in the next section, factor variables will be specified based on the distribution of labels, their proportions and possibly various kinds of relationships, including correlations, etc. In [19], an analysis of the dimensionality reduction of variables based on the study of the correlation between variables is also carried out, and it is proposed to consider as variables in machine learning models only the set of those variables for which the following restriction is valid

$$Data_{comprassed} = \left(x_i : \sup_{j \neq i} cor(x_i, x_j) < \rho^* \right)$$

where ρ^* is some correlation threshold (the authors use $\rho^* = 0.8$ when testing this approach). Reducing the dimensionality of tabular data in the example of tracking gene changes presented in the form of pictures was considered in [20], and the Manhattan distance between the pixels of the picture was considered as the distance between the pictures, regardless of the variation of the pixels of the picture. Thus, the authors of [20] propose to represent tabular data in vectors, matrices, and general tensors with a regular metric. Another approach to creating compact copies of data is considered in [21], where the concept of $(\lambda,\ \varepsilon)$ of a private budget is introduced, based on which the concept of proximity of two databases is built. Two sets of tabular data, $M(S)$ and $M(S')$, are called close with parameters $(\lambda,\ \varepsilon)$, if

$$D(M(S_1), M(S_2)) = \frac{1}{\lambda - 1} \log E_{x \sim M(S_1)} \left(\frac{P(M(S_1) \sim x)}{P(M(S_2) \sim x)} \right) \leq \varepsilon$$

Thus, the authors use the concept of sample likelihood ratio, which, on the one hand, simplifies the consideration of data similarity and, on the other hand, requires additional assumptions about the distribution of tabular data.

2. Related Works

Building compact yet informative dataset summaries and organizing corpora by similarity intersects several research threads: type-aware tabular representations, distributional distances for heterogeneous variables, learned representations for tables, corpus-level indexing/graphs, and privacy-aware data exchange. Below we outline the most relevant lines and position our work relative to them.

Compact representations for tabular data

A longstanding strategy is to replace raw data with concise, interpretable descriptors. For classical numeric variables, moment vectors (means, variances, higher moments) are ubiquitous because they support downstream statistical model- ing with minimal I/O. Recent systems formalize tabular structures and propose learned abstractions (e.g., DANets) that outperform several tree-based and deep baselines but typically assume access to full data and a known specification F [11]. SwitchTab extends neural autoencoding to tabular settings with competitive accuracy [15]; similarly, perturbation of autoencoder weights has been explored for compression and classification [13]. These learned approaches yield strong predictive embeddings but are less interpretable and less portable across tools than explicit summaries.

Orthogonal work compresses non-tabular modalities (e.g., images, shapes) via distributional or topological descriptors [6, 8, 10]. While methodologically insightful, these papers address different data structures. Our framework stays within tabular data yet adopts the same philosophy: summarize distributions with minimal assumptions and preserve the features that matter for comparison and modeling. Concretely, we define compact data representations (CDRs) per type: frequency tables (factor), fixed-bin histograms (time), moment vectors up to fourth order (numeric), and TF–IDF vectors (string) [25].

Similarity metrics over heterogeneous summaries

A second strand studies distances between distributions and summaries. For categorical distributions, Hellinger and total variation are standard; for histograms and scalar distributions, Wasserstein/earth mover's distance is widely used; and for vectors, L1/L2 norms and absolute/relative errors (MAE/MAPE) remain practical choices [26]. Distance correlation extends beyond linear dependence and has been proposed for representation and comparison tasks [17]. Our contribution here is not a single new metric but a *unified aggregation*: the Data Information Structural Similarity (DISS) score combines type-specific distances via a weight vector (user-set or learned), yielding a single number per dataset pair that respects heterogeneity and supports graph construction.

Representation learning for tables vs. interpretable summaries

Neural approaches for table understanding (e.g., [11, 15]) and "tables-to-images" transformations for CNNs [20] produce latent embeddings that are powerful for prediction. However, they (i) require full access to data, (ii) may entangle semantics across variable types, and (iii) are harder to audit. Our design favors *interpretable, type-aware* CDRs that can be exchanged across teams or accounts, inspected directly, and recomputed cheaply in pipelines without model retraining.

Corpus-level organization and graph structures

Graph-based representations are natural for encoding relationships between datasets. Prior work on graph databases provides schemas and storage models for complex, attributed graphs [16]. We instead operate one level higher: given pairwise DISS scores computed from CDRs, we build an adjacency graph over datasets and derive a minimum-spanning tree to obtain a similarity tree for navigation, nearest-neighbor retrieval, and hierarchical browsing. This differs from graph databases in that edges are induced by statistical similarity rather than pre-existing semantic relations.

Privacy-aware comparison and synthetic tabular data

When data cannot be freely shared, summarization and synthesis are alternatives. CTAB-GAN+ focuses on *generating* realistic tabular data for privacy-preserving use [21]; our objective is complementary—*comparing* datasets via summaries without exposing raw records. By exchanging CDRs (distributions, moments, TF–IDF vocabularies) rather than rows, organizations can perform cross-domain similarity assessment with reduced exposure of sensitive attributes, while still retaining practical utility for discovery and drift monitoring.

Positioning and gap

In summary, existing literature offers: (i) powerful learned embeddings for tabular prediction [11, 15, 13]; (ii) well- established distances for comparing distributions and vectors [26, 17]; (iii) graph-based storage models [16]; and (iv) syn- thesis for privacy [21]. What remains under-specified is a *unified, deployable pathway* that (a) constructs *interpretable, type-aware* compact representations from raw tables, (b) aggregates *heterogeneous* distances into a single, tunable similar- ity score at the dataset level, and (c) organizes entire corpora into a similarity graph/tree that is *scalable* and cloud-native. Our CDR+DISS framework addresses this gap and integrates directly with standard AWS tooling for large-scale, opera- tional use.

3. Information Technology Construction

3.1. Algorithm for Determining the Type of Variables

This section will propose the primary methods for creating compressed copies, taking into account the characteristics of the data and the models in which the data will be used. As shown in the literature review, the main approaches of constructing compressed copies are based on using moments of elements and various kinds of associations, such as the correlation matrix (6) and higher-order tensors. As practice shows, for a sufficiently accurate reproduction of the distribution of a random distribution, at least four-moment functions are required, i.e., in the presented (5), we can choose $k = 4$. Taking this fact into account, let us consider the representation of each of the column types, assuming that the sample x describes the column:

- Factor variables (binary, ordinal or order variable).

Main assumption for factor variables is the next

$$rel = \frac{\#\{unique\ values\ for\ x\}}{N} \leq Pcrit$$

where p_{crit} is a critical value for relation between number of unique values *# unique values* and number of observations *N*. If this relation is small, then with large probability corresponding variable is factor variable or numerical integer variable with small numbers of unique values. So, main sub-algorithm for factor variable identification is next

- ✧ Define value *rel* for variable;
- ✧ Check types of variables: if values are string, then variable is factor variable, if values are numerical, then check in the lists values of possible factor values (length, variable structure – twelve characters as in the card or 8 characters as in the UID), then variable is factor variable with high probability. In other cases, variable x is numeric (integer) with high probability;

- Time variables. This variables type can be checked using standard date checkers, which is represented in many software's. So, this type of variables is check using list of formats, for example *year.month.day* (*Y.m.d.*).
- Numerical (continuous and discrete variables): Numerical continuous variables can be characterized by few main properties:

  ✧ Numbers of unique values are co-dimensional with number of observations

$$\#\,\{\mathit{unique\ values\ for\ x}\} \approx N$$

or in more correct form

$$\frac{\#\{\mathit{unique\ values\ for\ x}\}}{N} \approx 1$$

  ✧ Continuous variables often are related to some distribution – for many financial actives this is normal distri- bution, log-normal distribution, heavy-tail distribution or mixtures;

- String variables. String variables, unlike factor variables (represented as strings) or various types of indicators (person id, platform id,. . .), are string variables with different lengths. Thus, it is most likely to have a string type (not a factor) if it is weakly structured (various record lengths) and the number of unique values is proportional to the number of records in the data, i.e.

$$0.1 \le \frac{\#\,\{\mathit{unique\ values\ for\ x}\}}{} \le 1.$$

3.2. Representing Different Types of Variables – Compression

In this section we consider compact representation of the data with different types? Described in the previous section.

- Represent factor variables with distribution as dictionary

$$Variable\colon \{values\colon \{val11,\ val12\},\ val11 : N_{val11},\ val12 : N_{val12}\};$$

This representation can be extended to any categorical variable with additional information for ordinal variables with strict order of the values.

- Date and time variables. If number of values for of date and time variables are small ($\le$ 1000), we can use the same representation as for factor variables. On other case (if number of observations are huge) it's possible to use frequency distribution with next representation of the date variable (date1, . . . , daten):

Intervals	$[D_1, D_2)$	$[D_2, D_3)$	...	$[D_k, D_{k+1})$
Frequencies	n_1	n_2	...	n_k

where

$$\min_i date_i = D_1 < D_2 < \ldots < D_{k+1} = \max_i date_i$$

Notice that for comparison, number of intervals must be fixed (for example $k = 100$). For this type of variables, we represent them as dictionary with next structure

$$Variable : \{separators : \{D_1, \ldots, D_{k+1}\}, Freq : \{n_1, \ldots, n_k\}\}$$

- Numeric variables. For numeric variables as in machine learning methods best way to use moments and number of observations. So, we can represent numerical samples by next vector

$$\left(N, \sum_{i=1}^{N} x_i, \sum_{i=1}^{N} x_i^2, \ldots, \sum_{i=1}^{N} x_i^k\right)$$

where k is some fixed value enough for numerical variable representation. In statistic often use $k = 4$, because first four moments are enough for understanding variable nature. If $\sum_{i=1}^{N} x_i^l$ is high, then we can use instead average value:

$$\sum_{i=1}^{N} x_i^l \to \frac{1}{N}\sum_{i=1}^{N} x_i^1$$

Hence, we can summarize numerical variable as

$$Variable1 : \left\{ moments : \left\{ N, \sum_{i=1}^{N} x_i, \sum_{i=1}^{N} x_i^2, ..., \sum_{i=1}^{N} x_i^k \right\} \right\}$$

- String variables. For this type we can use well-known vector representation of the string variables, for example as TF-IDF. In this case each variable can be represented as list of UNITS (all possible words or n-grams and corresponding sparse matrix. So, each variable for this type will be represented as

$$Variable1 : \left\{ \begin{array}{c} n\_gram\_length : n; \\ units : \{unit_1, unit_2, ..., unit_M\} \\ tfidf\_repres : SPARSE\_M\ ATRIX \end{array} \right\}$$

where *n_gram_length* is defined *n*-gram;

- Other characteristics. In many cases need to calculate additional numerical characteristics of the variables – correlation and/or covariance. This need for example is some mathematical models are used for prediction data of constricting forecast. In this case good assume is calculation of correlation or covariation matrix:

$$COR_{COV} = \{variables : \{var1,\ var2, \ldots, \},\ correlation : \{corr\},\ covariance : \{cov\}\}.$$

3.3. Comparing Compressed Data

Consider main metrics for comparison of the difference data types:

- Factor variables (binary, nominal and order). For this case we can use one of the following metrics

  ✧ Mahalanobis metric with identity matrix Σ, i.e.

$$d(p, q) = (p - q)^T \Sigma (p - q),$$

where $| * |_2$ is the standard Euclidian metric;

  ✧ Hellinger distance which can be defined by next relation

$$d(p,q) = \sqrt{\frac{1}{2}\sum_{i=1}^{n}\left(\sqrt{p_i} - \sqrt{p_2}\right)^2 (p_i + q_i)}$$

  ✧ Total variance distance which can be defined in the next way

$$d(p,q) = \sup_A |P(A) - Q(A)|$$

where measures P, Q related to the mass functions p, q corresponds.

  ✧ Wasserstein metric with parameter p = 1 which are defined in the next form

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- Consider now metric for frequencies tables for date variables:

Intervals	$[D_1, D_2)$	$[D_2, D_3)$	...	$[D_k, D_{k+1})$
Frequencies	n_1	n_2	...	n_k

and

Intervals	$[\tilde{D}_1, \tilde{D}_2)$	$[\tilde{D}_2, \tilde{D}_3)$	...	$[\tilde{D}_K, \tilde{D}_{K+1})$
Frequencies	$\tilde{n}_1$	$\tilde{n}_2$	...	$\tilde{n}_k$

Consider some function f with three properties

- $f(0) = 0$;
- f continuous and increase in R_+;
- $\lim_{x\to\infty} f(x) = \infty$;

Then for two frequencies tables, we define next metric

$$d(FR, \tilde{FR}) = \frac{1}{k}\sum_{i=1}^{k} f\left(\left|D_i - \tilde{D}_i\right| + \left|D_{i+1} - \tilde{D}_{i+1}\right| + \left|n_i - \tilde{n}_i\right|\right)$$

Good guess for this metric is function

$$f(x) = e^{\alpha x} - 1,$$

where $\alpha > 0$ is an additional parameter, which must be defined from test samples, differences between dates are defined in some time units (days, hours, . . .).

- For numeric variables, we can use relative difference between variables. Assume that we have two vectors

$$\left(N = \sum_{i=1}^{N} x_i^0, \sum_{i=1}^{N} x_i, \sum_{i=1}^{N} x_i^2, ..., \sum_{i=1}^{N} x_i^k,\right)$$

$$\left(M = \sum_{i=1}^{M} y_i^0, \sum_{i=1}^{M} y_i, \sum_{i=1}^{M} y_i^2, ..., \sum_{i=1}^{M} y_i^k,\right)$$

Then metric between this two vectors are defined as Euclidian metric or as MAPE or MAE.

- For string variables with Sparse matrix as main form of representation we can calculate frequencies of n-grams and use difference between two frequencies as proposed in factor variables section, where frequencies will be calculated as

$$q_i = \frac{\sum_{k=1}^{N} SPARSE\_MATRIX_{ki}}{\sum_{i=1}^{M}\sum_{k=1}^{N} SPARSE\_MATRIX_{ki}}$$

where *SPARSE MATRIX* is updated for two list of units (*units*1, *units*2)

- For correlation or covariance, we can use any matrix metric

For calculation difference between two datasets, we can use weighted metric:

$$DISS(Dataset1,\ Dataset2) = w_1 \cdot DIST(Factor\ variables) + w_2 \cdot DIST(Frequencies) + w_3 \cdot DIST(Numerical) + w_4 \cdot DIST(Other\ info = correlation,\ covariance\ldots),$$

where the vector of parameters $w = (w_1, w_2, w_3, w_4)$ is defined by some reason, for example by the model or variables type.

4. Information Technology Realization

In developing an algorithm for classifying and comparing variables in tabular data, a multi-stage information technology was created, running in the Amazon Web Services (AWS) cloud environment and built on Apache Spark [22, 23]. The goal of making the technology is to determine the types of variables, create a concise description for each of them, and perform a comparison of data structures between different sources or files. This approach allows you to fully automate the analysis of data structures in both research and applied tasks—for example, when checking data quality, combining information from multiple sources, or before preparing to train machine learning models. The technology is also adapted to real-world challenges—it considers large volumes of data, various input formats, and the need for repeatable and transparent calculations.

The data processing process is implemented as a PySpark application running on an Amazon EMR cluster [22]. The central part of the application is responsible for reading CSV files from S3 storage [24], cleaning the data (including removing missing data, extra characters, commas and spaces), and type matching. Determining the type of a variable—boolean, factorial, numeric, time or text—is based on the analysis of parameters such as the number of unique values, the length of the strings, or the ability to convert the value to a date. The technology can also process files without headers and data containing noise or partially corrupted records.

As a result, we will receive a concise and informative representation of each variable. For example, simple frequency tables are built for categorical and logical values, and for numbers, key statistical indicators are calculated: mean, variance, standard deviation, maximum, and minimum. If the variable refers to time, it is presented as a histogram divided into time intervals. Text fields are converted into numerical vectors using TF-IDF—an indicator that identifies essential words in the text [25]. This approach allows you to significantly reduce the amount of data for further processing without losing critical information.

The technology uses metrics selected according to the data type to compare variables. For numerical variables, the mean absolute error (MAE) or relative error (MAPE) is used, as well as distributions—measures such as Wasserstein distance, total variation, or Hellinger metric [26]. Text data represented via TF-IDF are estimated using L1 or L2-norm. Comparison of temporal variables is carried out on the basis of frequency histograms. The total difference between data sets is determined by calculating the weighted sum of metrics selected for each variable according to its type. This approach allows you to form a generalized similarity indicator—DISS (Data Information Structural Similarity), which can indicate the stability of the structure or the presence of anomalies in the data.

Scaling the developed solution and ensuring efficient work with large volumes of data is carried out using the AWS cloud infrastructure. All CSV files are stored in an S3 bucket with a structured organization—this makes it easy to identify the data's source, category, or period. In this architecture, S3 acts as a buffer for input data and a repository for processing results. In addition, archiving and version control mechanisms are implemented to facilitate convenient auditing and ensure full reproducibility of the analytical process.

All stages of data processing are managed using Apache Airflow [23]. This orchestrator automatically monitors the appearance of new objects in S3 and activates the appropriate processing. DAG processes sequentially perform key actions: create an Amazon EMR cluster, launch a PySpark application, wait for processing to complete, and write the results to S3. All operations are performed directly in the cloud, which allows you to manage the infrastructure centrally, provides easy scaling, and simplifies monitoring. Thanks to the detailed logging provided by Airflow, it is easy to detect potential errors, receive notifications about failures, and, if necessary, restart individual tasks.

After complete processing, the results are stored in JSON or Parquet formats on S3 [27]. The data structure in the generated files is registered in the AWS Glue catalogue [28], which allows you to access them through Amazon Athena

[29] using SQL queries instantly. This approach eliminates the need to create a separate repository or configure complex ETL processes—the data is immediately ready for further use.

As can be seen in Fig. 1, the information technology is built on a modular architecture that supports cloud-based scalable data processing [22]. The main components of the technology are:

Input data storage (Amazon S3): all input files in CSV format are uploaded to a specially created S3 bucket storage. The folder naming structure allows you to identify the data source, receipt date, and input file type. This part serves as a single entry point for the technology [24].

Process orchestration (Apache Airflow): automation of all processing stages is implemented through Apache Airflow [23]. Its DAG process performs the following tasks: detecting new objects in S3, initializing the EMR cluster, launching a computational task on the cluster, waiting for processing to complete, saving the results, and subsequently registering them in metadata.

Data processing (Apache Spark + Amazon EMR): the main computational logic is developed as a Spark application written in PySpark [22]. This application implements all the logic of a local Python analyzer (CSVAnalyzer) in a dis- tributed environment. Spark allows you to process many files in parallel and effectively scale the technology depending on the load – Fig. 2.

Aggregation and storage of results (Amazon Athena + AWS Glue): processing results (summaries by variables, types, vector representations) are stored in S3 in Parquet or JSON format [27]. Metadata is automatically registered in the AWS Glue Data Catalog [28], which allows you to query the results using SQL through Athena without first loading them into the database.

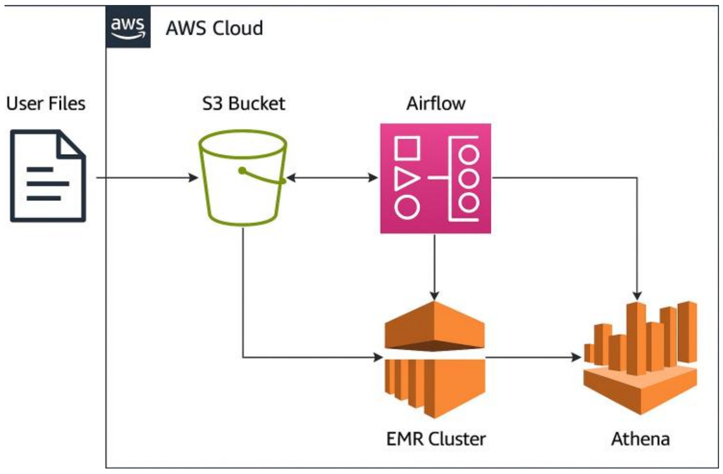

Fig.1. Technology architecture

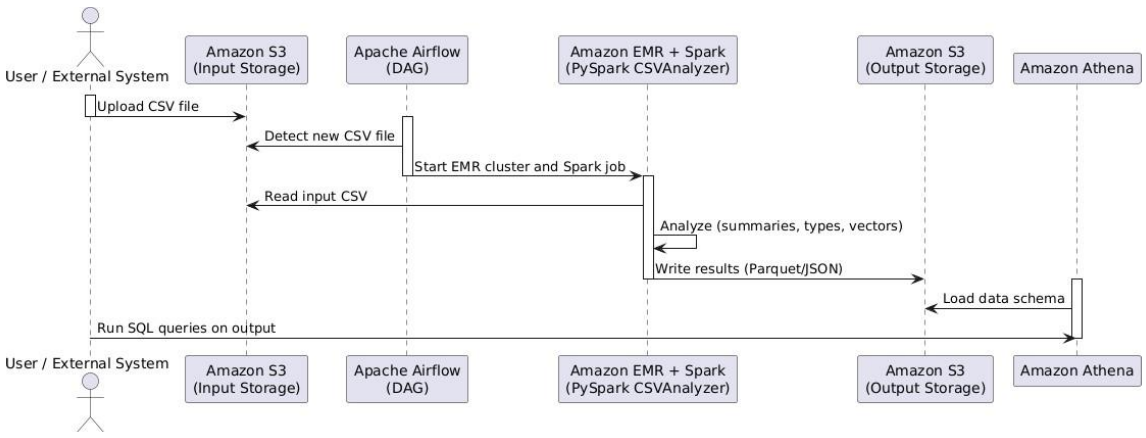
User / External System
Amazon S3 (Input Storage)
Apache Airflow (DAG)
Amazon EMR + Spark (PySpark CSVAnalyzer)
Amazon S3 (Output Storage)
Amazon Athena
Upload CSV file
Detect new CSV file
Start EMR cluster and Spark job
Read input CSV
Analyze (summaries, types, vectors)
Write results (Parquet/JSON)
Load data schema
Run SQL queries on output

Fig.2. Sequence diagram of the technology

In the first stage, the technology receives data from files (in our case, CSV files with trading data for Apple and Tesla) stored in S3 [30]. The content is read with or without column names, depending on the header format. Missing values, spaces, and extra characters are removed during preprocessing, and variables are typed. Each variable in the table is classified according to its statistical and structural properties. A variable can be assigned to one of the following types:

- Boolean variables (binary logical values);
- Factorial or categorical variables (limited number of unique values);
- Numeric variables (discrete and continuous);
- Time variables (values that can be parsed as dates/times);
- String variables (structured and unstructured, depending on variations in the length of records).

The technology forms a compact representation of the variable based on the identified type. For Boolean and factor variables, this is a frequency table; for numeric variables, a vector of moments (mean, variance, maximum, minimum); for time variables, a histogram with binning; for string variables, a TF-IDF matrix with a limited number of features [25]. All of these representations generalize the input data, allowing you to reduce the storage volume and quickly perform inter-set comparisons.

To compare variables from different data sets, a specialized module is used that supports several metrics: mean absolute error (MAE), relative error (MAPE), Wasserstein, Hellinger distance, total variation, L1/L2 matrix distance for TF-IDF, and frequency distance for binned time variables [26]. Aggregation results are formed as a weighted sum of metrics, considering weights for each variable type.

Integration, Scalability and Performance. The advantage of the implemented technology is its full integration with the AWS infrastructure: the use of standard services for orchestration (Airflow), processing (EMR), storage (S3), metadata catalogue (Glue) and queries (Athena) ensures high reliability, repeatability of calculations and ease of deployment. The result is stored on S3 in Parquet format, which can be used further in Athena.

Table 1. Compression performance of parquet and ORC

Dataset	Format	Original Size (CSV)	Compressed Size
Apple (daily)	Parquet	412 MB	87 MB
Tesla (daily)	Parquet	398 MB	114 MB
Apple (daily)	ORC	412 MB	79 MB
Tesla (daily)	ORC	398 MB	98 MB

As shown in Table 1, both Parquet and ORC achieve substantial reductions in storage size compared to the original CSV files. On average, ORC slightly outperforms Parquet in compression ratio, while Parquet remains competitive and widely adopted due to its integration with analytic frameworks such as Spark and Athena. These results illustrate the benefits of columnar formats for large-scale tabular data management.

5. Conclusions & Future Works

Developing methods for comparing tabular structured data based on their characteristics is a progressive direction in machine learning in general and big data in particular. A large number of works are devoted to the issue of dimensionality reduction and the selection of the main characteristics of data based on the use of well-known machine learning models, including classical works on dimensionality reduction (the principal component method), as well as more recent works on the study of data dynamics aimed at identifying structural changes in models based on data-driven approaches. Among the classical machine learning algorithms, it is worth noting that models for processing big data are based on representatives, for which only some numerical characteristics of data packets are essential and not the data packets themselves (CURE, BFR clustering algorithms, etc.). The primary attention of the work is devoted to models for constructing compressed copies of data, which, on the one hand, would most broadly reflect the essence and characteristics of the data required for mathematical models that will use these characteristics and, on the other hand, would not be too wide for their opti- mal storage. The literature review indicated that most of the works focus on probabilistic data characteristics since these characteristics are used for data analysis and model parameter estimation. Taking this fact into account, the paper consid- ered the theoretical and practical aspects of construction an information technology for construction compressed copies of data, methods for their comparison and construction a data similarity graph, which will further provide a good toolkit for analyzing data changes and classifying unknown data. In future work in this direction, it is planned to expand the functionality of the information technology, taking into account the theoretical justifications and the analysis of existing data corpora.

In our case studies on multi-source CSV corpora, the proposed pipeline—compact data representations (CDRs) coupled with the DISS similarity metric—reduced storage and I/O by replacing raw tables with small, interpretable summaries, yet preserved enough structure to recover the nearest related datasets and to assemble an intuitive similarity tree over the corpus. In practice, this made exploratory work faster and more reproducible: the end-to-end AWS realization (EMR/Spark for computation, S3/Parquet for storage, Glue/Athena for access, Airflow for orchestration) behaved like a standard, auditable data product that can be scheduled, logged, and re-run without bespoke infrastructure. These benefits come with trade-offs that are typical for summary-based methods. The approach favors interpretability, portability, and heterogeneity awareness—each variable type is summarized and compared with type-appropriate statistics and distances—at the cost of record-level fidelity and some sensitivity to modeling choices (bin edges for time histograms, vocabulary size for TF–IDF, and the relative weighting of types in DISS). In practice, we found these trade-offs man- ageable: rare or ambiguous cases can be escalated to row-level checks only for top candidates surfaced by DISS; limited multivariate structure can be reintroduced by adding a small set of cross-feature summaries (e.g., covariance or correlation blocks) where it matters; and stability improves by fixing global bin edges across the corpus, capping vocabularies with an OOV bucket, and normalizing or learning DISS weights rather than setting them ad hoc. From a theoretical standpoint, the summaries used here (empirical frequencies, fixed-edge histograms, TF–IDF with a fixed vocabulary, and low-order moments) converge to their population counterparts as sample sizes grow, and the underlying distances (Hellinger, total variation, Wasserstein, L1/L2, MAE/MAPE) inherit this convergence. As a result, DISS is permutation-invariant in row order and becomes stable with more data; for numeric variables, standardizing before moment calculation further im- proves invariance to linear rescaling. Computationally, CDR construction scales linearly in the number of rows, while pairwise comparisons over a corpus scale with the number of dataset pairs; for very large repositories, approximate k-NN graph construction or blocking by schema can retain the same workflow with subquadratic cost. In operational use, we recommend defaults that worked well across domains—moments up to fourth order for numeric columns, calendar- aligned fixed bins for time variables, TF–IDF over unigrams and bigrams with a capped vocabulary, and balanced type weights in DISS normalized by feature counts—then tuning only where labeled pairs or domain constraints justify it. Un- der these settings, the method proved effective for data discovery and cataloging, drift monitoring before model retraining, duplicate and near-duplicate detection during ingestion, and pragmatic dataset selection in MLOps pipelines, providing a practical bridge from raw tables to corpus-level structure without exposing raw records.

References

[1] R. Kannan, G. Bayraksan, and J. R. Luedtke, "Technical Note: Data-Driven Sample Average Approximation with Covariate Information", *Operations Research*, 2025.

[2] C.R. Siddharth and S.B. Vishwadeepak, "Enhancing Cloud Migration Efficiency with Automated Data Pipelines and AI-Driven Insights", *International Journal of Innovative Science and Research Technology*, vol. 9(11), 2025.

[3] U. Sarmah, P. Borah, and D. K. Bhattacharyya, "Ensemble Learning Methods: An Empirical Study", *SN COMPUT. SCI.*, vol. 5, 924, 2024.

[4] D. Zhang, C. Yin, J. Zeng, X. Yuan, and P. Zhang, "Combining structured and unstructured data for predictive models: a deep learning approach", *BMC Med Inform Decis Mak*, vol. 20, 280, 2020.

[5] J. Errasti, I. Amigo, and M. Villadangos, "Emotional Uses of Facebook and Twitter: Its Relation With Empathy, Narcissism, and Self-Esteem in Adolescence", *Psychol Rep*, vol. 120(6), pp. 997-1018, 2017.

[6] J. Cui, M. Cui, B. Xiao, and G. Li, "Compact and Discriminative Representation of Bag-of-Features", *Neurocom- puting*, vol. 169, 2015.

[7] M. J. Goswami, "Leveraging AI for Cost Efficiency and Optimized Cloud Resource Management", *International Journal of New Media Studies*, vol. 7(1), 2020.

[8] C. Zheng, R. Zheng, R. Wang, S. Zhao, and H. Bao, "A Compact Representation of Measured BRDFs Using Neural Processes", *ACM Transactions on Graphics*, vol. 41, pp. 1–15, 2022.

[9] E. Zohner, E. Gunning, G. Hooker, and J. Morris, "CLaRe: Compact near-lossless Latent Representations of High- Dimensional Object Data", 2025. 10.48550/arXiv.2502.07084.

[10] Z. Long, H. Meng, T. Li, and S. Li, "Compact geometric representation of qualitative directional knowledge", *Knowledge-Based Systems*, vol. 195, 105616, 2020.

[11] J. Chen, K. Liao, Y. Wan, D. Chen, and J. Wu, "DANets: Deep Abstract Networks for Tabular Data Classification and Regression", 2021. 10.48550/arXiv.2112.02962.

[12] K. Labunets, F. Massacci, F. Paci, S. Marczak, and F. Oliveira, "Model comprehension for security risk assessment: an empirical comparison of tabular vs. graphical representations", *Empirical Software Engineering*, vol. 2(6), pp. 3017–3056, 2017.

[13] S. Abrar, and M. Samad, "Perturbation of deep autoencoder weights for model compression and classification of tabular data", *Neural Networks*, vol. 156(C). pp. 160–169, 2022.

[14] C. Bordenave and B. Collins, "Strong asymptotic freeness for independent uniform variables on compact groups associated to nontrivial representations", *Inventiones mathematicae*, vol. 237, pp. 221–273, 2024.

[15] J. Wu, S. Chen, Q. Zhao, R. Sergazinov, C. Li, S. Liu, C. Zhao, T. Xie, H. Guo, C. Ji, D. Cociorva, and H. Brunzell, "SwitchTab: Switched Autoencoders Are Effective Tabular Learners", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 15924–15933, 2024.

[16] S. Á lvarez-Garc´ıa, B. Freire Castro, S. Ladra, and O. Pedreira, "Compact and Efficient Representation of General Graph Databases", *Knowledge and Information Systems*, vol. 60, pp. 1479–1510, 2019.

[17] X. Liang, Y. Qian, Q. Guo, and K. Zheng, "A data representation method using distance correlation", *Frontiers of Computer Science*, vol. 19, 191303, 2024.

[18] C. Troisemaine, J. Flocon-Cholet, S. Gosselin, S. Vaton, A. Reiffers-Masson, and V. Lemaire, "A Method for Discov- ering Novel Classes in Tabular Data", *IEEE International Conference on Knowledge Graph (ICKG)*, pp. 265–274, 2022.

[19] S.-K. Kim, "Compact Data Learning for Machine Learning Classification", *Axioms*, vol. 13, 137, 2024.

[20] Y. Zhu, T. Brettin, F. Xia, A. Partin, M. Shukla, H. Yoo, Y. Evrard, J. Doroshow, and R. Stevens, "Converting tabular data into images for deep learning with convolutional neural networks", *Scientific Reports*, vol. 11, 11325, 2021.

[21] Z. Zhao, A. Kunar, R. Birke, H. Scheer, and L. Chen, "CTAB-GAN+: enhancing tabular data synthesis", *Frontiers in big data*, vol. 6, 1296508, 2024.

[22] Amazon Web Services. Amazon EMR Developer Guide. *Amazon Web Services Documentation* 2023. Available online: https://docs.aws.amazon.com/emr/ (accessed on 19.05.2025)

[23] Apache Software Foundation. Apache Airflow Documentation. *Apache Airflow*. Available online: https://airflow.apache.org/docs/ (accessed on 19.05.2025).

[24] Amazon Web Services. Storage Best Practices for Data & Analytics. *AWS Whitepaper* 2022. Available on- line: https://docs.aws.amazon.com/whitepapers/latest/building-data-lakes/building-data-lake-aws.html (accessed on 19.05.2025).

[25] University of Illinois. Data Science Discovery: Data Types. *University of Illinois.* Available online: https://discovery.cs.illinois.edu/ (accessed on 19.05.2025).

[26] Mercadier, Y. Distance Measures for Probability Distributions. *Distancia Library Documentation* 2022. Available online: https://distancia.readthedocs.io/en/latest/ (accessed on 19.05.2025).

[27] Apache Software Foundation. Apache Parquet. *Apache Parquet*. Available online: https://parquet.apache.org/ (ac- cessed on 19.05.2025).

[28] Amazon Web Services. AWS Glue Documentation. *Amazon Web Services Documentation* 2023. Available online: https://docs.aws.amazon.com/glue/(accessed on 19.05.2025).

[29] Amazon Web Services. Amazon Athena Documentation. *Amazon Web Services Documentation* 2023. Available online: https://docs.aws.amazon.com/athena/ (accessed on 19.05.2025).

[30] M. Moazeni, "Automating Stock Market Data Pipeline with Airflow", Spark, Postgres. *Medium* 2023. Available on- line: https://medium.com/@mehran1414/automating-stock-market-data-pipeline-with-apache-airflow-minio-spark- and-postgres-b67f7379566a (accessed on 19.05.2025).

Authors' Profiles

Igor V. Malyk was born in Ukraine on July 19, 1983. In 2005 he graduated from Yuriy Fedkovych Chernivtsi National University (Chernivtsy, Ukraine) by specialty *statistics*. In 2009 he defended his PhD thesis at the Institute of Cybernetics (Kiev, Ukraine) by specialty *theoretical foundations of computer science and cybernetics*. In 2018 he defended his doctoral thesis at the Institute of Cybernetics (Kiev, Ukraine) by specialty *theoretical foundations of computer science and cybernetics*

From 2005 to 2017, he worked as an assistant, and later as an associate professor at the Department of Systems Analysis and Insurance and Financial Mathematics (Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine). Since September 2017, he has been working as an associate professor at the Department of Mathematical Problems of Control and Cybernetics (Yuriy Fedkovych Chernivtsi National University, Chernivtsi,

Ukraine), and since 2024, he has been a full professor and head of the department. Research interests are self-adaptive algorithms in machine learning, random processes, stochastic differential equations, deep learning, neural networks.

Prof. Malyk is membership of CHEST Journal editorial team (statistical reviewer); Multidisciplinary Digital Publishing Institute (MDPI) editorial team; American Mathematical Society and International Society editorial team for Computational Biology(ISCB).

Yevhen Kyrychenko was born in Ukraine on October 23, 1998. He received the B.Sc. and M.Sc. degrees in computer science from Ivan Franko National University of Lviv, Ukraine, in 2020 and 2022, respectively. He is currently pursuing the Ph.D. degree in software engineering at the Department of Software Engineering, Yuriy Fedkovych Chernivtsi National University, Ukraine. His major field of study includes cloud computing, big data technologies, and distributed systems.

He is currently working as a Senior Software Engineer at EPAM Systems, Ukraine. He has been involved in research and development projects related to cloud-native architectures, scalable data processing, and distributed computing. His research has been published in peer-reviewed conference proceedings. His current research interests include distributed data processing systems, big data analytics, and cloud infrastructure optimization.

Mykola Gorbatenko was born in the city of Chernivtsi, located in the Chernivtsi region of Ukraine. He received his Master's degree in Computer Science from the Faculty of Applied Mathematics at Chernivtsi National University, Ukraine, in 2005. In 2013, he was awarded the scientific degree of Candidate of Physical and Mathematical Sciences (specialty 01.05.04 – System Analysis and Theory of Optimal Solutions) by Taras Shevchenko National University of Kyiv, Ukraine.

He is an Associate Professor in the Department of Mathematical Modeling at Chernivtsi National University, Ukraine. His areas of interest include analysis, optimization, architecture, and high-load applications.

Taras Lukashiv was born in Ukraine on March 20, 1983. He received his Master's degree in Mathematics from the Faculty of Applied Mathematics at Yuriy Fedkovych Chernivtsi National University, Ukraine, in 2004. In 2010, he was awarded the scientific degree of Candidate of Physical and Mathematical Sciences (specialty 01.05.02 – Mathematical Modelling and Calculation Methods) by Yuriy Fedkovych Chernivtsi National University, Ukraine.

Since 2004, he worked as an assistant professor, and from 2019 to 2024 as an associate professor at Yuriy Fedkovych Chernivtsi National University.

From 2022 to 2024 - visitor scientist at the Luxembourg Institute of Health, Luxembourg. From 2024 - post-doctoral researcher at the University of Luxembourg, Luxembourg Centre for Systems Biomedicine, Department of Clinical and Translational Informatics.

Dr. Lukashiv is membership Multidisciplinary Digital Publishing Institute (MDPI) editorial team, Journal of Economics and Management Sciences editorial team, and Natural & Mathematical Sciences in Medicine and Medical Education editorial team; American Mathematical Society, International Society for Computational Biology, and International Society for Clinical Biostatistics.

How to cite this paper: Igor V. Malyk, Yevhen Kyrychenko, Mykola Gorbatenko, Taras Lukashiv, "Data Optimization through Compression Methods Using Information Technology", International Journal of Information Technology and Computer Science(IJITCS), Vol.17, No.5, pp.84-99, 2025. DOI:10.5815/ijitcs.2025.05.07