LUND UNIVERSITY

SCHOOL OF ECONOMICS AND MANAGEMENT

Semantic Similarity in Data Mesh Environments: A Column Classification Approach

Analiz Delgado Medina & Denisse Cortez Budnik

Lund University School of Economics and Management

Supervisor: Hamed Sabahno

DABN01 - Data Analytics and Business Economics: Master Essay I

Seminar date: June 2, 2025

CONTENTS

Semantic Similarity in Data Mesh Environments: A Column Classification Approach

Analiz Delgado Medina & Denisse Cortez Budnik

June 2, 2025

Abstract

Efficient classification techniques are needed in decentralized environments due to the rapid increase in high-dimensional and semi-structured data in recent years. In this thesis, we investigate the usability of text embeddings for column classification by analyzing semantic distance metrics to detect semantic similarities between different database columns. `OpenAI-text-embeddings-3-small` model has been used to calculate four different distance metrics: Cosine, Euclidean, Manhattan and Chebyshev to measure the distances between embeddings in vectorial space. Furthermore, a clustering analysis has been conducted to further validate the classification performance. For a better refinement of the data a comparison between two different dimensionality reduction techniques has been implemented with three different clustering models, respectively. Our results depict cosine distance maintains a high performance in line with previous literature. Additionally, clustering implementation correctly corroborates the relationships obtained by the similarity distance metrics.

Keywords: text embeddings, column classification, semantic similarity, distance metrics, clustering analysis

Acknowledgments

We want to express our gratitude to Adage for trusting us by providing this thesis case and for relying on our knowledge to work on it. We would like to thank Måns Wirfelt, who dedicated his time from the very beginning to explain the company's objectives in this thesis case and tirelessly shared his knowledge and time with us. We would also like to thank Hamed Sabahno for supervising us, for his hands-on experience, his time and patience in answering our questions and sharing his technical and practical knowledge on the subject.

In addition, we sincerely appreciate the support of our friends in the DABE cohort and others; their advice, encouragement, and presence made all the difference throughout this journey. In the most stressful moments, their laughter echoing through the basement was not only a source of joy but also a reminder of the strength found in their friendship. Thank you for all the unforgettable memories; they will stay with us forever.

1 INTRODUCTION

Within the framework of data mesh and federated analytics a general column classifier is developed by intersecting multiple fields of entity resolution, metadata management, schema matching and machine learning. This chapter reviews existing research and business procedures that highlight their applicability to the issue at hand. We will examine the strengths and weaknesses of the various strategies and explore deeper the gaps that this thesis seeks to fill.

Data mesh is an emerging paradigm in data management that overcomes the drawbacks of decentralized data architectures. It shifts the focus from centralized data lakes and warehouses to domain-oriented ownership. Based on four fundamental concepts it was first introduced by Dehghani (2020), in the article "How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh" and are the following:

1. Domain-oriented decentralized data ownership: encourages accountability and domain-specific expertise by allocating data ownership to specific business domains.
2. Data as a product: data should be handled with the same attention to detail and quality as a customer-facing product.
3. Self-serve data infrastructure: giving independent teams the resources and the infrastructure needed to manage their data products.
4. Federated computational governance: establishing a governance framework that strikes a balance between centralized policies with decentralized execution to guarantee compliance and interoperability.

Data mesh enables organizations to scale data management effectively, mitigating the disadvantages associated with conventional centralized data architectures. Domain-oriented teams are directly responsible for the ownership of their data products, guaranteeing higher data quality since they are accountable for the creation, upkeep, security and refresh of the data. This approach for decentralized governance minimizes discrepancies and enhances data management. It advocates a federated governance approach that reinforces uniform policies and security practices among all domains while allowing flexibility in local execution.

A further relevant area in our research is schema matching. Schema matching involves recognizing analogous attributes among various datasets, a foundational concept in column classification. The study on schema matching titled "Survey of approaches to automatic schema matching" (Rahm and Bernstein, 2001) investigates the techniques used to align similar attributes within diverse datasets. Column classification can be viewed as a form of schema alignment, in which it is essential to assess if a new column is semantically similar to existing ones. Some traditional schema matching techniques depend on:

- String-based similarity: utilizing string distance metrics to compare column names.

- Rule-based matching: utilizing predefined rules (e.g., "if a column contains "ID", it serves as an identifier").
- Ontology-driven matching: employing external ontologies to comprehend semantic connections among attributes (Shvaiko and Euzenat, 2013).

These methods facilitate the unification of data from various sources by aligning their structures, which is crucial for enabling data sharing and interaction between different systems and/or platforms. Maintaining consistency in data schemas helps preserve data accuracy and minimize errors. It improves interoperability of applications and databases facilitating the merging or comparison of datasets from different sources, especially beneficial in settings where data comes from multiple heterogeneous systems (Forster, 2024). Nonetheless, they have potential limitations, a few of which are:

- Reliance on column names: columns having distinct names yet similar content that remain unmatched (e.g., "Client ID" vs. "User ID").
- Rigidness: rule-based systems require manual updates.
- Limited generalization: ontology-driven methods need well-defined taxonomies, which may not exist for many real-world datasets.

Recent studies integrate machine learning and deep learning (Ebraheem et al., 2017) for schema matching, showing how embeddings and similarity learning enhance automated classification. It has been demonstrated that probabilistic models can classify columns according to statistical patterns and distributions and column profiling is an essential part of feature extraction for ML-based column classification. Therefore, a well designed profiler can generate metadata that improves classification accuracy. Additionally, embedding-based methods have demonstrated that word embeddings can enhance column similarity detection. Additional methods utilized in deep learning with transformers like BERT-style models have reached cutting-edge accuracy in metadata classification.

Research Aims

In light of these challenges and previous research undertaken, the objective of this thesis is to focus on utilizing embeddings for column classification. We will analyze text data and examine various distance similarity metrics to conduct comparisons and assess the effectiveness of column classification. The research is organized around developing a model for the utilization of any individual user within the company Adage, aiming for these objectives:

Using Natural Language Processing (NLP) tools: we will create text embeddings using OpenAI tools. This will enable us to transform text data into vector representations of high-dimensional data that reflects semantic significance.

Examining Distance Similarities: we will investigate different distance similarity metrics, including Cosine similarity, Euclidean distance, Manhattan distance and Chebyshev distance, to assess their efficiency in column classification. This examination will aid us in

comprehending how various metrics affect classification effectiveness of embedded textual data.

Clustering analysis: along with similarity metrics, we will perform a clustering analysis to enhance our results. This will entail categorizing data points to evaluate if the outcomes generated from similarity metrics are robust and consistent. Additionally, clustering analysis offers a visual representation of the data, facilitating a clearer comprehension of the connections and classification of text columns and detecting the presence of outliers or anomalies.

The inquiries directing this investigation are:

- Are embedding-based representations of text data a feasible method for column classification?
- What similarity measurement technique offers the greatest accuracy for text classification utilizing embeddings?
- Is it possible for the incorporation of clustering analysis to enhance the accuracy of classification for embedded text data?

The rest of the thesis is structured as follows: Section 2 summarizes previous literature. Section 3 presents the data. Section 4 discusses the empirical strategy. Section 5 presents the results. Section 6 presents a detailed analysis on the results. Section 7 presents the results for the two additional Databases. Section 8 digs deeper into the characteristics of each dataset and its impact on the clustering method. Section 9 concludes. Section 10 presents suggestions and comments on future work that could be done.

2 LITERATURE REVIEW

In recent years, the increase of high-dimensional and semi-structured data has inspired studies on efficient methods for column classification, particularly within the realm of data mesh and federated systems. In this section of our thesis, we will examine current studies and industry methods that emphasize their relevance to the issue being addressed. We will examine their strengths and weaknesses and explore further the issues that this thesis seeks to tackle.

The rise of large amounts of high-dimensional data in companies has made traditional concepts like data warehouses and data lakes challenging to manage and utilize effectively. Li and Toor (2024) explored how alternative approaches such as data mesh promote a federated architecture, where data ownership remains with the domain that generates it. While federated learning is a related concept in machine learning, the main idea is federated data architecture, which supports decentralized ownership and processing without moving raw data across domains.

As data management seamlessly transitions towards decentralized paradigms like data mesh, the need for properly and efficiently classifying textual data becomes increasingly important. Conventional classification methods, frequently rely on bag-of-words or handcrafted rules, face challenges with scalability and semantic generalization. This has resulted in the extensive use of word embeddings, which enables more nuanced semantically representations of textual information.

Embeddings such as those derived from word2vec, doc2vec, or contextual models, transform discrete tokens into continuous vector spaces. This allows for the semantic similarity to be captured and computed more effectively. Li, Saihan and Gong, Bing (2021) showed that embeddings greatly improve text classification tasks by offering dense and semantically rich input to deep learning models. This enrichment translates to the ability to carry meaningful connections between the words. Their tests on different architectures (e.g., CNN, GRU, LSTM, and others) showed that models utilizing pre-trained word2vec embeddings, especially a two-layer GRU model, delivered better performance, indicating the significant importance of embedding duality and structure in classification accuracy downstream.

Jin et al. (2016) expanded the skip-gram model to present a bag-of-embeddings approach designed for text classification. They suggested developing multi-prototype word embeddings—unique embeddings for every word based on text class labels—to grasp subtle context-sensitive meanings. This approach presumes that each word can demonstrate varying semantic characteristics based on the column or document category in which it is found. By optimizing the likelihood of these embeddings specific to each class, their method surpassed multiple traditional and neural benchmarks on both balanced and imbalanced datasets. This discovery reinforces the feasibility of employing class-aware embeddings in column classification situations where context and domain semantics differ significantly.

Building on the idea that there are embeddings specific to each class, various embedding techniques have been developed to enhance text representation and classification effi-

ciently. da Costa et al. (2023), focused on analyzing three of these techniques –Word2Vec, GloVe, BERT–. Word2Vec is one of the most used text classification techniques; literature demonstrated a high performance by this model. GloVe's approach is similar to that of Word2Vec, they both present a word using a high dimension vector and are trained according to the surrounding words. The differences lie in the use of matrices to embed the word context, which GloVe used but Word2Vec does not. BERT was another embedding technique discussed, it consists of a neural network architecture, which contains a big amount of parameters.

Further validating these findings, Alvarez and Bast (2017) examined how Word2Vec reinforces the similarity between words when they are found in a similar context. This similarity came about by the spatial distance that they share. The shorter the spatial distance is, the more similar the words are. GloVe and Word2Vec presented a different idea within the same concept, with a notable exception in the training, since GloVe is faster trained. GloVe distinguished itself by using statistics integrated into a similar vector space as Word2Vec. The statistical measure used is the "co-occurrence probability".

While embeddings can capture relationships within textual data, the effectiveness of classification tasks heavily depends on the similarity metrics employed. A study by Gómez and Vázquez (2022) evaluated various distance metrics in combination with embedding techniques—such as word2vec, fastText, GloVe, doc2vec, and ELMo—to assess document similarity.

They found that performance varied significantly depending on the metric used, with each embedding exhibiting strengths with specific measures. Doc2vec paired with cosine similarity yields the most consistent results overall. However, it is important to note how inconsistencies arise when handling very large documents for the combination of embedding and distance metric. BERT was excluded from their analysis due to its high computational cost.

Their evaluation also identified doc2vec embedding with cosine similarity and soft cosine similarity as an effective use of similarity measurements, nevertheless some important limitations in cosine similarity consistency are found. Steck et al. (2024) critically pointed out that using cosine similarity could lead to inconsistencies. Their analysis indicated that cosine similarity could produce inconsistent outcomes based on the embedding training goal and regularization. In particular, they argued that the measure is not consistent under certain rescaling of embeddings and may not accurately reflect true semantic closeness when the embeddings' norms fluctuate unpredictably. This observation is especially important for column classification systems that depend on retrieval or matching based on similarity between column names and their descriptions.

Although cosine similarity is popular for its ease of use and scale independence, the results from both studies highlight a warning: similarity measures might act unpredictably based on the distribution and training of embeddings. Moreover, they conducted an empirical evaluation comparing the use of similarity measures and found that the results vary depending on the choice of embedding, computational constraint and contextual representation. In light of this, the present thesis will undertake an empirical assessment of various similarity-based approaches, analyzing their effectiveness and dependability for this specific classification task.

While embeddings enhance text representation and classifications across NLP tasks, their applications expand further than general text analysis. Recent studies have shown the efficiency of text embeddings for unsupervised column classification in data management systems. The seminal research by Hulsebos et al. (2019) in Sherlock system developed a model for semantic type detection that accurately classifies columns, integrating GloVe and FastText (unsupervised word embeddings) along with statistical attributes to depict column semantics. Although mainly intended as a supervised classifier, Sherlock's embedding structure implicitly confirmed that text-oriented vector representations can encapsulate semantic connections among columns, like recognizing alike categories.

Expanding on this, Hou et al. (2020) introduced Column2Vec that targets unsupervised column classification by systematically analyzing distance metrics. Their research showed that cosine similarity is more effective than euclidean distance for text columns, whereas jaccard similarity is superior for categorical data. The threshold-based clustering method demonstrated encouraging outcomes in organizing columns without predetermined types, although this technique is restricted to single-table scenarios and does not address cross-database schema matching issues. Both studies depend on static word embeddings and fail to thoroughly investigate the capabilities of contextual embeddings or consider performance on messy, real-world data.

While Column2Vec improves unsupervised classification through distance metric analysis, embedding-based clustering can become difficult to interpret due to the high dimensionality of vector representations. Upadhye (2023a), proposed the application of dimensionality reduction techniques like UMAP. In addition, the use of hyperparameters when using UMAP are crucial to generating low dimensional embeddings. The study dictated the importance of correctly tuning Nearest Neighbors and Minimum Distance when using UMAP to find a balance between detailed neighborhood structures or global structures.

Once UMAP has been applied to reduce the dimensionality of the embeddings, clustering techniques can be used to uncover unseen patterns in the data. This investigation evaluated how the use of three clustering techniques—K-Means, Agglomerative clustering, and HDBSCAN—contributes to the results by tackling data with different characteristics.

K-Means remained as one of the most popular and simple clustering algorithms, even though this technique was published over 50 years ago (Jain, 2010). Ahmed et al. (2020), who also studied this methodology, commented on how Wu et al. (2008), had listed k-means as one of the top 10 best clustering techniques for data analytics. Jain (2010) credited its long-lasting implementation to the simplicity of use of this technique as well as its successful application results.

Agglomerative clustering, on the other hand, began by considering each of the data points as its own cluster; slowly, it merges together the two most similar clusters until the entirety of the dataset is in one cluster (Emmendorfer and de Paula Canuto, 2021). This technique was guided by a linkage method; the linkage determines the distance between the points in the vector space by linking the points together to form the clusters (Tokuda et al., 2022). Reimers et al. (2019) also found that when agglomerative clustering is applied to highly argumentative data, the results are better than K-Means and DBSCAN. In particular, they argued that agglomerative clustering achieves a better performance when combined with average linkage. This pairing enhances cluster formation, particularly

when working with embedded data.

Clustering of short text can be challenging since it contains noisy data. DBSCAN was a popular clustering technique to cluster noisy data, as expressed by Asyaky and Mandala (2021a). HDBSCAN improved on this technique by being able to handle clusters with varying densities, unlike DBSCAN, which struggled with this (Saha, 2023a). Embedding short text data with FastText, paired with UMAP for dimension reduction and HDBSCAN for clustering, performed the best according to NMI and purity scores.

Together, these studies underscore significant opportunities for enhancing unsupervised column classification, such as exploring contextual embedding models, creating hybrid similarity metrics for diverse data types, and assessing clustering algorithms. The existing challenges in managing cross-database situations and imperfect data offer especially important avenues for research aimed at creating stronger column classification systems.

Our thesis focuses on textual data obtained from data columns like descriptions or sampled values, an increasingly crucial form of metadata in federated analytics and data mesh contexts. While previous studies have investigated word- and document-level embeddings for conventional text classification, this thesis focuses specifically on the task of automatically classifying column semantics solely using textual information. For this purpose, OpenAI's embedding models will be utilized to create vector representations of column text.

The goal is to evaluate the effectiveness of various distance metrics in column classification. Additionally, complementing the study with a cluster analysis that will enhance results by categorizing data points and assessing the robustness of similarity metrics. The implementation of these techniques will facilitate improved metadata management and data discovery in decentralized settings.

3 DATA

Data is the key to understanding how embedding techniques work and how they can improve semantic similarity and classification. The databases chosen for this thesis were provided by Adage due to their vast knowledge in this field. These databases provide a real-world representation of enterprise data from both short text and NLP data. This coincides with our reasoning behind the investigation, creating a base for the use of embeddings to enhance data organization and accessibility in companies.

To better understand our data, this section will be divided into three parts:

- Description of data: An overview of the databases, their background, and their relevance to the thesis. This section will emphasize Database A as our primary database as well as Database B and C for validation of findings.
- Data preprocessing: The methodology applied to clean and transform the data which will be converted to embeddings, semantic similarity, classification and clustering.
- Data limitations: A discussion on the challenges encountered while using the databases, including factors that may affect model performance.

The following table provides a summary of the databases used in this thesis. It outlines their source, how many datasets compose them, and their purpose of use for this analysis. It highlights the differences between each of the databases, which will be discussed further in the following subsections.

Database Names	Source	Datasets Contained	Purpose of use
Database A	Microsoft	9	Primary dataset
Database B	Kaggle	7	Validate findings
Database C	Microsoft	25	Validate findings

Table 1: Overview of Databases Used

3.1 ***Description of data***

3.1.1 Database A

The primary database—Database A, corresponds to Microsoft's AdventureWorks database—serves as the first exploration towards the usage of embeddings on this type of data. Mitri (2015) describes AdventureWorks as an excellent tool for educational and research purposes, particularly in database management and machine learning applications. Designed as an Online Transaction Processing (OLTP) system, which imitates business transactions, offering a structured and comprehensive dataset.

Database A, which consists of nine interconnected datasets, primarily focusing on sales and product information. The data warehouse component emphasized by Microsoft

in regards to this database, enhances the database's usability, This structured design support the belief of a preexisting relationship between the datasets located inside of Database A, which touch on the usability of semantic similarity for text classification This interconnectivity between these datasets may enables the use of embeddings and support advanced analyses of semantic similarity distances. Because of the structure of the database, it also provides an ideal foundation for clustering techniques, helping to uncover hidden relationships within the data as well as a better understanding of it.

Database A was originally created to imitate the operations of a bicycle manufacturing company, providing a real-world representation of the processes seen by these types of businesses. It contains data spanning multiple years of sales records, customer details, product categories, subcategories, descriptions, returns, and calendar-based trends. Because this database provides a range of information, it facilitates a practical approach to training machine learning models on realistic business processes.

Our goal is for the results of this thesis to be applied in an enterprise setting. Because of this, the use of AdventureWorks as our primary database is highly suitable. Its simulation of real-world applications, this database allows for the model to process information from a baseline behavior, serving as a reliable reference for assessing classification and semantic similarity algorithms when applied to other datasets. However, given that this thesis will focus on textual data, AdventureWorks does present a limitation. Due to its high proportion of numerical data, two additional text-based datasets will also be analyzed to corroborate the results of Database A and enhance the depth of the analysis.

3.1.2 Database B

Database B is formed by seven datasets; it complements Database A by introducing diverse forms of textual content. This database's data was recompiled from Kaggle; it was built with the purpose of applying the model to a textual-based dataset. Because of the diverse origins of the datasets, the data may consist of real-world applications or simulated text data.

The datasets consist of the following information:

- Amazon sales, sales of different Amazon product categories over time.
- Bank additional full, marketing campaign of a Portuguese banking institution.
- Customer-Churn-Records, customer characteristics, and behavior that could lead to churn.
- Dkhousing, sampling housing prices in Denmark.
- Ecommerce, operations of an online retail company.
- Supermarket sales, growth in the sales of supermarket companies.
- World bank dataset, simulation of World Bank metrics used for policymaking, forecasting, and analysis.

3.1.3 Database C

Contoso is a fictional international conglomerate foreseeing the sale of more than one hundred thousand products. It was built on the idea of representing real-world data. With a workforce of over twenty-seven thousand employees, Contoso is a leading organization in the manufacturing, sales and support industry with their headquarters and IT center located in Paris.

The global presence that Contoso has may hinder the searchability and access to documents. The company's current solution is the use of VPNs by their employees to gain access to other office's information. Because this has posed an issue, the company aims to fast-track its digital transformation by using cloud services to connect employees, partners, data, and workflows. This will consequently allow the company to maintain its competitive status in the market while fostering data-driven solutions.

While Contoso's global presence brings advantages, they have also led to collaborative issues among departments due to inconsistent security protocols and inconsistencies across databases. These enterprise challenges present a unique opportunity of simulating a realistic business environment, allowing the model to be tested and evaluate its capabilities of enhancing data integrity.

The use of Contoso's database – Database C – enables the model to evaluate its ability to identify enterprise data as well as measure its effectiveness in analyzing corporate data. The database includes sales trends, customer behavior, product information, financial transactions, and supply chain logistics, all of which will undergo embedding techniques. By mapping the data into vector space, the model will analyze proximity between vectors to determine their semantic similarity, which will enable column classification.

Transforming Database C into embeddings and calculating their semantic similarity will enable seamless data searchability across departments, which will improve connectivity throughout Contoso. This similarity classification approach enhances data accessibility, ensuring employee access to related information from across the company.

Data accessibility from all offices will allow Contoso to conduct informed decision-making and trend detection in customer behavior and production activities. The use of embedding models allows for an automated approach to data integration, reducing the need for manual data merging as well as a greater data consistency across datasets.

3.2 ***Data preprocessing***

To ensure consistency throughout the data, preprocessing was applied. File encoding and delimiter were implemented on each dataset to correctly interpret the data. Encoding was automatically detected using Chardet minimizing potential encoding errors and reducing the need for manual intervention. Delimiters were determined based on the frequency of common separating characters.

Standardization was applied to ensure uniformity for future dataset use. All file delimiters are changed to ",", and the datasets were encoded in UTF-8 encoding to maintain

uniformity in the database. Processed datasets are saved with their name and the suffix "_processed" which will allow subsequent preprocessing to skip over the unprocessed data, avoiding redundancy.

The datasets provided contain inconsistent encodings and delimiters, which can threaten the usefulness of the data, leading to misinterpretation. To ensure compatibility across different software environments, UTF-8 encoding was selected, along with a standardized comma "," delimiter. The updated file names help track modifications made to the original dataset. In addition, the model has been programmed to use the files whose suffix is "_processed" for future work.

Textual extraction was performed to isolate non-numerical data, ensuring compatibility with the model. An empty list was initialized to store column names that contain text. Each previously processed file –both csv files as well as excel files– pass through a text extraction pipeline, where textual columns are identified and added to the list. The same process applies to object-type data. Because data/time information can appear as an object, the code specifies that these columns should be excluded.

Once identified, textual columns are extracted and stored in datasets containing exclusively textual data. These datasets are saved in a designated output directory, facilitating targeted textual analysis. To maintain a consistent formatting and ensure compatibility between the datasets and code, all extracted datasets are saved in CSV format. The filenames receive the prefix "text_" followed by the original dataset name. The code ignores the appearance of missing values (NAs) to allow for real-world data practice, since this is the data commonly found in companies, allowing for a practical application of the model.

The text-exclusive data is divided into two segments, splitting the dataset at the midpoint to evaluate the consistency of embedding distances between the two halves. This division allows the model to assess whether embeddings preserve the relationship of the data across the different sections. For accurate file reading, delimiter detection is implemented using Sniffer, which automatically identifies the file's delimiter. If Sniffer fails, the system defaults to a counting method, this is done to ensure a vigorous base when standardizing delimiters.

From the first half of the dataset, one third of the information is randomly selected, keeping in mind that no segment should exceed 2000 rows, this subset is designated as "sample". This process is repeated for a second time using the second segment of the data, forming a subset called "compare". These partitions allow for a structured analysis of embedding consistency between both sections.

To streamline file organization and future analysis, separate directories are created for storing the sample and comparing datasets. For a proper tracking of the converted data, each file is named with its respective suffix, "_sample" or "_compare".

These new samples will allow for a direct comparison of embedding distances between the two dataset segments. A correctly functioning embedding model should be able to find a close similarity metric between the column it is classifying and the column it is identified as. This closeness is determined by the relative distance between them in vector space.

3.3 ***Data limitations***

Database C contains overlapping column information in some of its datasets, which can potentially introduce some challenges to the model – for example, *"RegionCountryName"* information also corresponds to *"SalesTerritoryCountry"* and *"ContinentName"* aligns with *"SalesTerritoryGroup"*. Given that the information in the columns contain duplicate values, the similarity metrics can confuse the two column names.

Dates are not identified by the preprocessing code as a numerical value nor as a date/time value. Although this problem can be solved by identifying the use of "/" to mean a date/time, this is not possible due to the use of "/" as a delimiter by certain datasets. Similarly, the erroneous detection of delimiters can lead to the misalignment of columns. Sniffer has problems finding the correct delimiters for certain columns, leading to a fall back count method being used in case of this failure.

Because the "$" symbol is viewed as a character, monetary values are kept in the datasets. Potential preprocessing fixes such as stripping the datasets of non numeric characters are available to fix this issue. However, it is important to note that this can also strip other information with necessary non-numeric characters –for example, stripping the "@" in emails–.

The exclusion of numerical data diminishes the amount of usable columns per data set. This is especially prevalent in Database A. The AdventureWorks data contains a big amount of sales data containing numerical data, nevertheless this data is compensated by the big amount of textual data contained in the datasets *"Customer"* and *"Products"*.

Although one-hot encoding was initially considered as a preprocessing step for categorical data, its use was ultimately excluded as it contradicts the fundamental objective of this thesis. One-hot encoding assigns strict categorical identities without preserving relationships between features, whereas embeddings allow for similarity-based identification of columns. Given that the thesis aims to classify columns based on similarity metrics derived from their embeddings in vector space, one-hot encoding would jeopardize the structural integrity of the columns, which ultimately serve as the data to be embedded.

4 METHODOLOGY

With the purpose of evaluating the different distance similarity metrics we have done empirical analysis over the previous datasets utilizing text embedding techniques offered by OpenAI. Our aim is to derive semantically relevant representations of the data and use these representations for the subsequent column classification analysis and posterior clustering analysis. If the evaluation demonstrates to have low distance scores for every category or column name, it will indicate that columns are accurately matched.

4.1 ***Sampling Model***

Before embedding the text data, we have experimented with various sample sizes, specifically 10, 50, 100 and 200, to assess the trade-off between computational cost and efficiency of the analysis. Based on these tests, and having observed the results, we concluded that a sample size of 50 provides the most balanced and representative results for our datasets. While larger sample sizes yield more robust and precise outcomes, the size of 50 offers a good compromise between performance and efficiency.

Once the samples are drawn, we proceed to apply the text embedding model to transform the textual data into numerical format. Embeddings allow each text element to be represented as an array of numbers, therefore capturing its semantic meaning and conceptual relationships in a vector space. This numerical representation enhances the usability of the data in many tasks like clustering and similarity measurement, not only classification, as the model now works on structured data rather than raw text.

4.2 ***Embedding Model***

In order to depict the text within a high-dimensional semantic space, we utilize OpenAI's text-embedding-3-small model. Adage allocated a budget for utilizing this advanced AI capability in our project. This model belongs to the third generation embedding series and offers dense vector representations of text tailored for semantic similarity and retrieval purposes. It accommodates variable-length inputs and produces fixed-length 1,536 dimensional embeddings. Each text input ti is encoded as an embedding $v_i \in \mathbb{R}^{1536}$.

The application of text-embedding-3-small is based on distributional semantics, which suggests that words and phrases found in similar contexts typically have similar meanings (Harris, 1954). The model undergoes training and it optimizes the following classification and clustering tasks. The meaning behind each embedding is encoded in such a way that subsequent tasks can depend on geometric connections between text vectors to determine similarity or dissimilarity.

To evaluate the similarity between text embedding, the following distance and similarity metrics will be analyzed. Each one of them provides a different perspective on how the proximity and similarity can determine column classification.

4.3 ***Similarities and Distances***

4.3.1 Cosine Distance

Cosine similarity measures the cosine of the angle between two non-zero vectors in an inner product space. It is defined as the direction rather than magnitude, explained by Manning et al. (2008) in the *"Introduction to Information Retrieval",* mathematically expressed as:

$$\text{CosineSimilarity}(x, y) = \frac{x \cdot y}{\|x\|\|y\|}, \tag{1}$$

where the numerator represents the *dot product* of the vectors vi, vj, while the denominator is the product of their Euclidean lengths. The effect of the denominator is to length-normalize the vectors to unit vectors, and the metric's range goes from [-1,1], where:

- 1: indicates identical orientation (maximum similarity).
- 0: indicates orthogonality (no similarity).
- -1: indicates opposite orientation (maximum dissimilarity).

Cosine distance is the complement of cosine similarity, it can be a more effective indicator than cosine similarity as it transforms this measure into a range from [0,1] for non-negative vectors. For general vectors this range extends to [0,2]. It is commonly used in clustering and nearest-neighbor searches when a distance measure (instead of a similarity score) is needed. Mathematically expressed as:

$$\text{CosineDistance}(x, y) = 1 - \text{CosineSimilarity}(x, y), \tag{2}$$

where:

- 0: indicates most similar vectors.
- 1: indicates no correlation.
- 2: indicates most dissimilar vectors, pointing in opposite directions.

This metric maintains the same benefits as cosine similarity but with a distance-based interpretation. Therefore, this is the one we will be taking into consideration in our analysis.

4.3.2 Euclidean Distance (L2 Distance)

Euclidean distance is a measure that quantifies the straight-line (L2) distance between two vectors in the high-dimensional space. Following Aggarwal et al. (2001), it is determined by:

$$\text{EuclideanDistance}(x_i, y_i) = \sqrt{\sum_{k=1}^{d} (x_i - y_i)^2}, \tag{3}$$

where d is the dimensionality of the embeddings. This metric is very sensitive to the scale of vectors and dimensionality and it is especially effective when the input space is normalized. It is important to highlight then, that the range of Euclidean distance values has no strict upper limit unless the inputs are normalized. For this reason, we have normalized them, after taking the square root of the vectors difference we get that for both unit vectors in R^n the maximum distance is $sqrt(2)$, the angle ranges within [0,1.414].

This metric will be especially beneficial in our clustering analysis in posterior techniques like K-Means and UMAP, where centroids are computed in euclidean space.

4.3.3 Manhattan Distance(L1 Norm)

Manhattan distance, also known as taxicab or city/block distance calculates the sum of absolute differences in all dimensions. Mathematically following Michel Marie Deza (2013):

$$\text{ManhattanDistance}(x_i, y_i) = \sum_{k=1}^{d} |x_i - y_i|. \tag{4}$$

It is more resistant to outliers than euclidean distance and is beneficial when embedding dimensions represent independent or non/interacting features. It can detect variations even when only a limited number of features differ significantly. In this case, the range depends on dimensionality and the distribution of the components on the space vector. The minimum value the lower bound takes is 0, however there is no fixed upper bound, as it depends on the number of components. A higher Manhattan distance indicates a greater separation between points in vector space.

4.3.4 Chebyshev Distance

Chebyshev distance measures the maximum absolute difference between any single dimension; it is particularly beneficial when a significant variance in a single attribute is more significant than cumulative differences, (Michel Marie Deza, 2013):

$$\text{ChebyshevDistance}(x_i, y_i) = \max |x_i - y_i|. \tag{5}$$

Chebyshev distance is seldom utilized by itself in NLP, but it can offer insights into feature-specific anomalies or serve as a component of combined distance metrics in anomaly detection systems. This metric slightly depends on how the components change coordinate-wise and also on dimensionality. Its minimum value is 0 whereas its maximum value will depend on the number of dimensions.

4.4 ***Dimensionality Reduction***

To complement the evaluation of similarity metrics a clustering analysis will be implemented. However, before addressing this analysis it is necessary to deal with the importance of dimensionality reduction. High-dimensional data frequently presents difficulties such as computational complexity, overfitting and the "curse of dimensionality" where the data space's volume grows exponentially with every new dimension, resulting in sparsity and complicating the analysis.

Techniques for dimensionality reduction seek to address these issues by transforming the data into a lower-dimensional space while preserving as much of the original information as possible. This procedure not only simplifies the data but also improves the effectiveness and interpretability of posterior analyses.

Dimensionality reduction techniques can be generally classified into linear and non-linear techniques. Linear approaches like Principal Component Analysis (PCA), presume that the data lies on a linear space and projects it into a lower-dimensional space by maximizing the variance. Formally, PCA solves the eigenvalue decomposition of the data covariance matrix, and it projects it onto the k eigenvectors to finally obtain the reduced dimension representation. Jolliffe (2002). Mathematically it is expressed as:

1. Given multivariate data the covariance matrix is computed:

$$S_{jk} = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k), \tag{6}$$

where

- S_{jk} is the entry in the j-th row and k-th column of the sample covariance matrix.
- x_{ij} is the value of the j-th variable for the i-th observation.
- $\bar{x}_j$ is the sample mean of the j-th variable.

The formula computes the covariance between variables j and k over n observations.

Although, it is not written in matrix notation as $\mathbf{C} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$, this is equivalent given centered data.

2. Eigenvalue decomposition: Principal components are obtained from the eigenvectors of the covariance matrix:

$$\mathbf{S}\mathbf{w} = \lambda\mathbf{w}, \tag{7}$$

where $\mathbf{w}$ is an eigenvector and λ is the associated eigenvalue.

3. Principal Component Scores: The data is finally projected onto the top-k principal components. Each component score for a sample $\mathbf{x}_i$ is:

$$z_i = \mathbf{x}_i^\top \mathbf{w}. \tag{8}$$

This is consistent with the formula:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}_k, \tag{9}$$

where

- $\mathbf{Z} \in \mathbb{R}^{n\times k}$: reduced-dimensional representation.
- $\mathbf{X} \in \mathbb{R}^{n\times d}$: centered data matrix with n samples and d features.
- $\mathbf{W}_k$: matrix of the top k eigenvectors (principal components).

PCA is widely used due to its ease of use and its ability to effectively capture linear associations in the data, nonetheless, it may not be suitable for complex data structures or non-linear data. Therefore, this dimensionality reduction technique will be carried out to complement UMAP.

Following that concept, non-linear techniques like t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) aim to maintain the local structure of the data while reducing dimensionality. t-SNE emphasizes preserving pairwise relationships among data points, whereas UMAP enhances a low-dimensional representation retaining the global structure of the data. For the development of dimensionality reduction in our analysis, UMAP is chosen as it retains both local and global structure of the data and its outstanding performance with large datasets, unlike PCA and t-SNE, this algorithm is optimized for performance and it is suitable for visualizing high-dimensional data in 2D or 3D.

Formally, UMAP attends to use a mathematical structure akin to a k-neighbor graph to approximate a manifold must follow a basic structure, following McInnes et al. (2018):

1. Constructs a weighted k-graph where the edge weights between points i and j are defined using a fuzzy set membership function:

$$\mu_{ij} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma_i}\right), \tag{10}$$

where σ_i is a local connectivity parameter for point i.

2. Applies some tranformation on the edges to ambient local distance. The low-dimensional embedding $\mathbf{Y}$ is obtained by minimizing the following cross-entropy loss between high-dimensional and low-dimensional fuzzy sets:

$$\mathcal{L} = \sum_{(i,j)} \mu_{ij} \log\left(\frac{\mu_{ij}}{\nu_{ij}}\right) + (1-\mu_{ij}) \log\left(\frac{1-\mu_{ij}}{1-\nu_{ij}}\right), \tag{11}$$

where ν_{ij} represents the edge weight in the low-dimensional fuzzy set based on distances in the embedding space.

Through the use of dimensionality reduction methods, we are able to transform the embeddings into a more manageable and understandable format, aiding the implementation of clustering algorithms. This is an essential step for ensuring that the posterior clustering analysis accurately reflects the underlying structure of the data and provides valuable insights into the semantic relationships of the columns.

4.5 ***Clustering analysis***

To assess the inherent structure present in the embeddings, various unsupervised clustering techniques were utilized, K-Means Clustering, Agglomerative Clustering and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). These methods have assisted in assessing whether the model designed effectively clusters similar columns according to its semantic significance, by grouping columns that belong to for instance: personal details (e.g., *"FirstName", "LastName"*) or financial information (e.g., *"AnnualIncome", "TaxAmount"*).

4.5.1 K-means clustering

This technique segments the dataset into k unique, non-overlapping clusters. The algorithm iteratively allocates data points to the closest cluster centroid and adjusts these centroids until the procedure stabilizes and converges. Mathematically proved following Arthur and Vassilvitskii (2007):

Given a dataset $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ and a desired number of clusters k, the goal of k-means is to find a set of centers $C = \{c_1, c_2, \ldots, c_k\}$ that minimizes the following potential function:

$$\phi = \sum_{x \in X} \min_{c \in C} \|x - c\|^2. \tag{12}$$

This function represents the sum of squared Euclidean distances between each data point and the nearest cluster center. The aim is to partition the data into k clusters such that this sum is minimized.

The algorithm operates iteratively through the following steps:

1. **Initialization:** Randomly select k initial centers $C = \{c_1, c_2, \ldots, c_k\}$ from the dataset.
2. **Assignment Step:** Assign each data point $x \in X$ to the nearest center c_i, forming clusters C_i.
3. **Update Step:** Recalculate each center c_i as the mean of all points assigned to cluster C_i:
$$c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x. \tag{13}$$
4. **Convergence Check:** Repeat steps 2 and 3 until the centers no longer change significantly, indicating convergence.

It's important to note that while this algorithm is widely used due to its simplicity and efficiency, it can converge to local minima, and its performance heavily depends on the initial placement of centers. The drawback that presents is that the number of clusters should be predefined and can be sensitive to initialization of the centroids, which may result in less effective clustering.

4.5.2 Agglomerative clustering

Agglomerative clustering is a hierarchical technique that constructs a nested structure of clusters by consecutively merging the nearest pair of data points or clusters Müllner (2011). Beginning with each data point as an individual cluster, the algorithm gradually combines them according to a distance metric until all points from one cluster or until a stopping criterion is achieved, forming a dendogram that represents the hierarchical clustering structure. Formally expressed as Müllner (2011) states:

1. Input: Let: $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ be a dataset with n points in d-dimensional space.

2. Initialization: Start with each point in its own cluster: $\mathcal{C} = \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\}$. Compute the initial pairwise distances between all points, for example using Euclidean distance:
$$D(x_i, x_j) = \|x_i - x_j\|_2. \tag{14}$$

3. Iterative Merging: At each iteration:

- Find the pair of clusters $A, B \in \mathcal{C}$ with the smallest distance:
$$(A^*, B^*) = \arg\min_{A \neq B} D(A, B). \tag{15}$$
- Merge them: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{A^*, B^*\} \cup \{A^* \cup B^*\}$.

- Update the distance matrix using the Lance–Williams formula:

$$D(A \cup B, C) = \alpha_A D(A, C) + \alpha_B D(B, C) + \beta D(A, B) + \gamma |D(A, C) - D(B, C)|. \quad (16)$$

4. Linkage Criteria: The values of $\alpha_A, \alpha_B, \beta, \gamma$ depend on the linkage method, down below we explain in detail the Average linkage type and Ward's method, both popular in agglomerative clustering:

Linkage Type	α_A	α_B	β	γ
Average (UPGMA)	$\frac{n_A}{n_A+n_B}$	$\frac{n_B}{n_A+n_B}$	0	0
Ward's	see below	see below	see below	0

For Ward's method, the distance between clusters is defined as the increase in within-cluster variance:

$$D(A, B) = \frac{n_A n_B}{n_A + n_B} \|\mu_A - \mu_B\|^2, \quad (17)$$

where μ_A and μ_B are the centroids of clusters A and B, and n_A, n_B are their sizes.

5. Stopping Criterion: Stop when all points are in a single cluster (complete hierarchy), or when having reached a predefined number of clusters k, by cutting the dendrogram at the appropriate level.

Unlike K-means clustering, this method does not need the number of clusters to be predefined and can capture complex data patterns. However, one main drawback is that it may require substantial computational resources, particularly with large datasets and the selection of the distance metric and linkage criterion, can have a big influence on the outcomes.

4.5.3 HDBSCAN

This technique is an enhanced form of DBSCAN that detects clusters by analyzing the density of the data points. It transforms DBSCAN into a hierarchical clustering method and it derives a clustering technique based on cluster stability Campello et al. (2013) . Mathematically:

1. Core Distance: For a data point x_i, the core distance is defined as the distance to its k-th nearest neighbor:

$$d_{core}(x_p) = \text{distance to the } m_{pts}\text{-nearest neighbor of } x_p. \quad (18)$$

2. Mutual Reachability Distance: The mutual reachability distance between two points x_i and x_j is:

$$d_{\text{mreach}}(x_p, x_q) = \max\left\{d_{\text{core}}(x_p),\ d_{\text{core}}(x_q),\ d(x_p, x_q)\right\}. \quad (19)$$

3. Minimum Spanning Tree (MST): Construct a weighted graph on the dataset using $\text{mreach}_k(x_i, x_j)$ as edge weights, and compute its minimum spanning tree (MST).

4. Cluster Hierarchy: Create a hierarchy of clusters by removing edges from the MST in order of decreasing weight (i.e., increasing density).

5. Cluster Stability: For a cluster C, its stability is defined as:

$$\text{Stability}(C) = \sum_{x_i \in C} \left(\lambda_{\text{end}}(x_i) - \lambda_{\text{start}}(x_i)\right), \tag{20}$$

where $\lambda = \frac{1}{\text{mreach}_k}$ represents the inverse density, and $\lambda_{\text{start}}(x_i)$ and $\lambda_{\text{end}}(x_i)$ denote the birth and death of point x_i in the cluster hierarchy.

6. Cluster Selection: The most stable clusters are selected by analyzing the condensed cluster tree, keeping those with the highest stability scores.

This technique does not need a predefined number of clusters and can accommodate clusters of varying densities and shapes. It is important to highlight that it also successfully detects noise points that do not belong to any cluster, making it robust to outliers.

To consolidate the previous concepts, utilizing these clustering techniques on the embeddings obtained, will allow us to evaluate the model's capacity to capture semantic relationships and classify columns, as well as providing a comparison between the different methods. The results will provide information about the model's success in clustering similar columns and uncover any possible weaknesses that could lead to further improvement. Furthermore, the quality of clustering will be assessed by the purity, NMI and silhouette score.

4.6 ***Hyperparameter tuning***

Finally, a crucial component for the analysis is to perform hyperparameter tuning, adjusting the hyperparameters that determine the behaviour of the clustering algorithms. Hyperparameters are not derived from the data but are defined prior to the learning process. Proper tuning is necessary to ensure that clustering outcomes are meaningful and align with the structure of the data.

Additionally, Grid Search was used due to the high-dimensionality of the hyperparameter space; to identify which hyperparameter combination leads to the optimal clustering configurations. Although it requires more computing time when the parameter space is larger, one of the greatest advantages that Grid Search presents is that it is straightforward, simple and comprehensive within a specified range, guaranteeing that all candidate settings are fairly compared.

For this analysis, we performed Grid Search over the different clustering algorithm's parameters, evaluating each candidate by using the silhouette score (Rousseeuw, 1987). This metric quantifies cluster separation and cohesion by measuring how similar each point is to its own cluster compared to neighbor clusters. It ranges from [-1,1]; poor

clustering to ideal clustering. The optimization process goes as follows:

1. Parameter Grids

- **K-Means & Agglomerative Clustering**: Tested cluster counts $k \in [2, 10]$.
- **HDBSCAN + UMAP**: Sampled combinations of:
  - UMAP hyperparameters (`n_neighbors`, `min_dist`, `n_components`)
  - HDBSCAN hyperparameters (`min_cluster_size`, `metric`, etc.)

2. Evaluation For each configuration:

- Embeddings were clustered
- The silhouette score was computed on the full dataset
- Invalid HDBSCAN results (e.g., all points labeled as noise) were discarded

3. Selection Criterion The configuration with the highest silhouette score was retained for each algorithm, chosen for its interpretability and applicability in unsupervised tasks and its scale-invariance, that also aligns with our use of normalized embeddings. Considering all the above, our focus will be on using UMAP for dimensionality reduction and the three different clustering techniques, therefore the hyperparameters to be tuned are the following:

4.6.1 UMAP hyperparameters

For UMAP several important hyperparameters influence the structure of the reduced embedding space:

- **Number of neighbors (*n_neighbors*)**: This parameter controls the equilibrium between preserving local and global structures. A greater value of *n_neighbors* promotes the retention of the overall manifold structure, allowing the capture of wider patterns within the data. On the other hand, a lower value emphasizes local structure, which can assist in recognizing detailed clusters.
- **Minimum distance (*min_dist*)**: This parameter defines the smallest distance between points in the low-dimensional representation. A smaller *min_dist* value brings clusters closer together, making them more compact but possibly sacrificing some overall structure. Conversely, a greater *min_dist* value enables increased separation between clusters, maintaining the global structure while potentially resulting in less distinct clusters.
- **Number of components (*n_components*)**: This parameter determines the number of dimensions in the reduced embedding space. It is usually set at 2 or 3 dimensions for visual clarity. Higher values of *n_components* can capture more information but can make visualization harder to interpret.

4.6.2 K-Means hyperparameters

- **Number of clusters (*n_clusters*)**: This algorithm requires careful tuning of the most important hyperparameter *n_clusters (k)*, which affects performance, cluster granularity, and quality. It is typically chosen using methods such as the elbow method or silhouette score—the approach we will follow in this analysis. While Euclidean distance is the standard for this method, other metrics such as Manhattan or Cosine distance can be applied, although they may deviate from the default formulation and potentially compromise convergence to the optimal result.

4.6.3 Agglomerative hyperparameters

- **Number of clusters (*n_clusters*)**: This hyperparameter is crucial and defines where the hierarchical clustering process will stop. Alternatively, this hyperparameter could be predefined by specifying a distance threshold to cut the dendrogram at a specific similarity level.
- **Metric (*metric*)**: This hyperparameter defines how pairwise similarity distances are computed, in conjunction with the chosen linkage criterion. In this case, cosine distance will be used to optimize the clustering models.
- **Linkage criterion (*linkage*)**: This hyperparameter determines how distances between clusters are calculated during the hierarchical merging process. It directly affects the shape and granularity of the resulting clusters. Common strategies include single, complete, average, and Ward's linkage. In this case, we choose *average* linkage as it provides a balanced approach on cohesion and separation of the clusters.

4.6.4 HDBSCAN hyperparameters

- **Minimum cluster size (*min_cluster_size*)**: This parameter specifies the smallest allowable size for a cluster. Increasing *min_cluster_size* results in fewer but larger clusters, in contrast, decreasing it enables the identification of smaller, more detailed clusters. This parameter is essential for managing the level of detail we are seeking of the clustering outcomes. Unlike the other methods, the great advantage that HDBSCAN specifically for unsupervised settings, where the number of semantic groups is unknown, is that it does not require predefining the number of clusters.
- **Metric *(metric)***: This hyperparameter defines how pairwise similarity distances are computed. In this case, euclidean distance on normalized UMAP space approximates cosine, therefore euclidean will be used to optimize the clustering models.
- **Cluster selection method *(cluster_selection_method)***: This hyperparameter determines how clusters are selected from the hierarchy:
  - **eom (Excess of Mass)**: favors more stable clusters.
  - **leaf**: selects leaf nodes, producing smaller clusters.

5 RESULTS

This section represents the key findings of this thesis, focusing on how embeddings and similarity metrics accurately classify column names based on the shortest similarity distance. After applying dimension reduction to the embeddings, an additional analysis was conducted using HDBSCAN clustering technique to identify cohesion within the similarity metric results. The results are given for Database A, as mentioned in Section 3, our primary database.

To evaluate semantic similarity, pairwise comparisons were performed using four different methodologies: Cosine, Euclidean, Manhattan and Chebyshev distance. These measurements help quantify the semantic closeness between column names, revealing patterns that lead to column classification.

The clusters reveal that column embeddings naturally form clusters based on the data patterns, providing further evidence to evaluate the effective use of embedding for column classification. PCA was initially considered as a dimensionality reduction technique; UMAP demonstrates better performance when it comes to preservation of the data structure while minimizing computational cost. Hence, UMAP was selected as the optimal method for dimensionality reduction, and PCA was omitted from the results section. The combination of the best-performing clustering technique, and UMAP allows for enhanced cluster definition and precise grouping.

5.1 ***Distances***

5.1.1 Cosine Distance

Table 2 below showcases the results for the cosine distance similarity metrics, where distances closer to 0 demonstrate to have a strong semantic similarity, therefore are possible matches, while those close to 1 represent dissimilar pairs:

Table 2: Cosine distances with strong semantic similarity between Sampled and Compared column names using OpenAI `text-embedding-3-small`.

Sampled Column Name	**Compared Column Name**	**Cosine Distance**
AnnualIncome	AnnualIncome	0.053218
BirthDate	BirthDate	0.073604
EducationLevel	EducationLevel	0.055498
EmailAddress	EmailAddress	0.121764
Gender	Gender	0.139239
HomeOwner	HomeOwner	0.036425
LastName	LastName	0.226146
MaritalStatus	MaritalStatus	0.089053
Occupation	Occupation	0.035788
Prefix	Prefix	0.130678

The results obtained show that exact or almost-exact column names matches such as *"AnnualIncome", "Birthdate"* and *"EducationLevel"* yield very close cosine distances; all the above have distance values <0.05. This validates the model's performance and competence in identifying identical or very similar column names. Therefore, there is a strong semantic similarity between the compared and sampled column names.

In contrast, Table 3 shows the results obtained for semantically unrelated pairs, for instance, *"Occupation"* and *"OrderDate"* have a cosine distance of 0.721044, *"Education-Level"* and *"OrderDate"* a distance of 0.715046, *"EducationLevel"* and *"AnnualIncome"* a distance of 0.712242, and *"HomeOwner"* and *"ProductColor"* a distance of 0.708659. They exhibit larger values, confirming the model's ability to discriminate between unrelated columns.

Table 3: Cosine distances with weak semantic similarity between Sampled and Compared column names using OpenAI `text-embedding-3-small`.

Sampled Column Name	**Compared Column Name**	**Cosine Distance**
Occupation	OrderDate	0.721044
EducationLevel	OrderDate	0.715046
EducationLevel	AnnualIncome	0.712242
HomeOwner	ProductColor	0.708659

These findings align with the theoretical foundations of cosine similarity in high-dimensional data as explained by Manning et al. (2008), embeddings efficiently group semantic related features despite having little textual overlap.

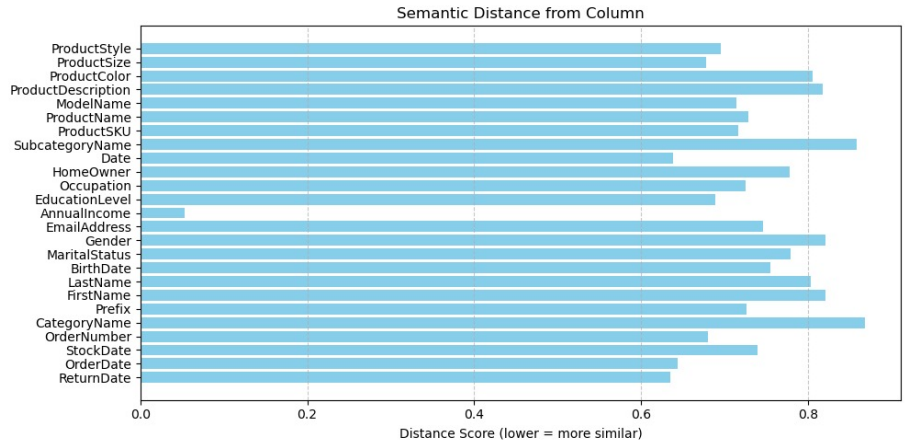

Figure 1: Cosine Semantic Distance from column "AnnualIncome"

Figure 1 shows the semantic distances displayed for each column relative to a reference column. In this case, we have represented the distances compared relative to the column *"AnnualIncome"* which is the one that has the lowest distance score, therefore,

the strongest semantic similarity. Additionally, it is visible that *"HomeOwner"* and *"OrderDate"* exhibit the lowest distance values, whereas, *"CategoryName"* and *"SubcategoryName"* present the highest scores, meaning that they have minimal semantic similarity. This representation facilitates visually classifying the columns.

5.1.2 Euclidean Distance

Table 4 displays the results for euclidean distance similarity metrics. Same as before, it can be seen that the following have the most semantically aligned column pairs. Some examples are: *"AnnualIncome"*, *"BirthDate"* and *"EducationLevel"*. These distances have values <0.4, confirming a strong classification within the correct column name.

Table 4: Euclidean distances with strong semantic similarity between Sampled and Compared Column Name using `text-embedding-3-small`

Sampled Column Name	**Compared Column Name**	**Euclidean Distance**
AnnualIncome	AnnualIncome	0.326245
BirthDate	BirthDate	0.383677
EducationLevel	EducationLevel	0.333160
EmailAddress	EmailAddress	0.493485
FirstName	FirstName	0.733935
Gender	Gender	0.527710

On the contrary, Table 5 presents the distances with the weakest semantic similarities, which means the ones with distance scores >1.414. These results make it clear that the model distinguishes between incorrect or misleading column matches.

Table 5: Euclidean distances with weak semantic similarity between Sampled and Compared Column Name using `text-embedding-3-small`

Sampled Column Name	**Compared Column Name**	**Euclidean Distance**
Gender	OrderNumber	1.251685
Gender	Occupation	1.247099
Gender	EducationLevel	1.245098
Gender	ProductSKU	1.242255

Additionally, Figure 2 displays the semantic distance towards the column *"AnnualIncome"* which has the lowest distance score in this case. The previous results are easier to interpret, it can be seen that *"SubcategoryName"*, *"CategoryName"*, and *"Gender"* have higher distances than 1.3 with this column, meaning they are more semantically different than others. Finally, most show a similarity distance that falls within [0.9,1.2] which suggests moderate semantic similarity.

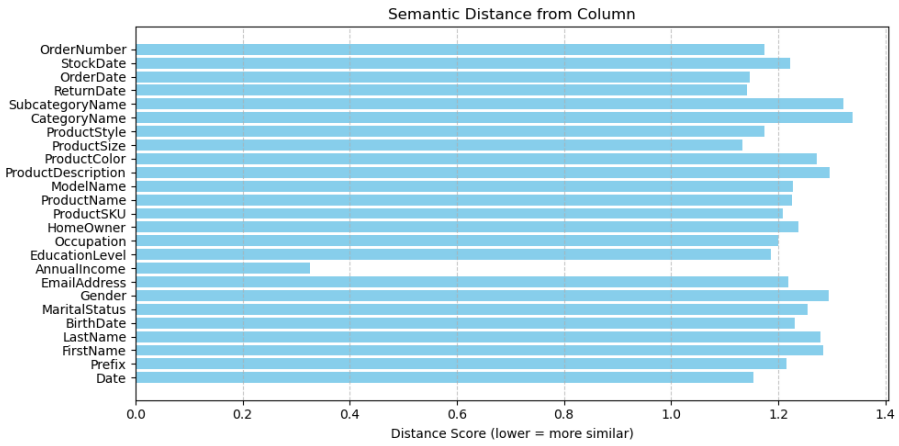

Figure 2: Semantic Distance from column "AnnualIncome"

5.1.3 Manhattan Distance

Table 6 provides a demonstration of the Manhattan distance, as discussed in Section 4. It lists the "Sampled Column Name" paired with the "Compared Column Name" that has the smallest distance between them, indicating the similarities between the embeddings. As explained previously, a lower Manhattan distance means that the compared columns are similar; this idea is shown on the Table by highlighting the closest matches.

Table 6: Manhattan distances with strong semantic similarity between Sampled and Compared Column Name using `text-embedding-3-small`

Sampled Column Name	**Compared Column Name**	**Distance**
AnnualIncome	AnnualIncome	10.111054
BirthDate	BirthDate	13.417226
EducationLevel	EducationLevel	8.872961
EmailAddress	EmailAddress	13.343703
FirstName	FirstName	20.784212
Gender	Gender	7.871207
HomeOwner	HomeOwner	19.780419
LastName	LastName	20.994518
MaritalStatus	MaritalStatus	7.711270
Occupation	Occupation	8.926117
Prefix	Prefix	5.015200

The results indicate that all sampled columns from Database A have the smallest Manhattan distance with the same column name from the compared subsection. The columns *"Prefix", "MaritalStatus"* and *"Gender"* reflect the smallest Manhattan distance out of all the columns with all their distances being < 8 units.

Table 7 contains the columns with the largest Manhattan distance. Each pair has a distance greater than > 38 units, indicating a significant separation between the Sampled and Compared columns. This table shows the values used by the model to differentiate between different columns.

Table 7: Manhattan distances with weak similarity between Sampled and Compared Column Name using `text-embedding-3-small`

Sampled Column Name	**Compared Column Name**	**Manhattan Distance**
HomeOwner	OrderDate	39.226682
HomeOwner	OrderNumber	39.071410
HomeOwner	OrderNumber	39.065589
Gender	ProductSize	38.720682

As shown in Table 7, certain columns have a stronger similarity due to their smaller Manhattan distance. Observing *"Prefix"* as having the shortest Manhattan distance, we pick this column to further analyze and determine the closest semantic similarities in the sampled and compared columns.

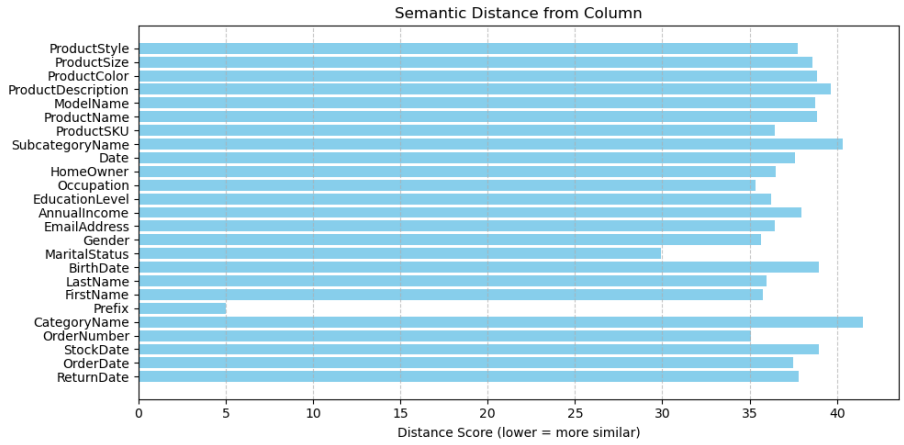

Figure 3: Semantic Distance from column "Prefix"

Figure 3 shows that the smallest distance shown in *"Prefix"* is with the comparison column of itself, the second being *"MaritalStatus"* and the third *"OrderNumber"*. The largest distances can be found in the columns *"SubcategoryName"* and *"CategoryName"*, both of them having a distance > 40 units.

5.1.4 Chebyshev Distance

By analyzing the Chebyshev distance, we can assess the computed distances for different columns and observe variations across columns with higher and lower distances. In Table 8 we will present the smallest observed distance of each of the columns in Database A.

Table 8: Chebyshev distances with strong semantic similarity between Sampled and Compared Column Name using `text-embedding-3-small`

Sampled Column Name	Compare Column Name	Chebyshev Distance
AnnualIncome	AnnualIncome	0.035255
BirthDate	BirthDate	0.042832
EducationLevel	EducationLevel	0.025310
EmailAddress	EmailAddress	0.045869
FirstName	FirstName	0.071211
Gender	Gender	0.027614
HomeOwner	HomeOwner	0.071302
LastName	LastName	0.072424
MaritalStatus	MaritalStatus	0.020445
Occupation	Occupation	0.043117
Prefix	Prefix	0.014443

Above, we highlight each of the columns in Database A along with their most similar column embedding, based on the smallest Chebyshev distance. Table 8 presents how each of the columns has the smallest distance within its comparison column. The column names with the biggest similarity are *"Prefix"*, *"MaritalStatus"* and *"EducationLevel"*, the three have a distance < 0.026.

Furthermore, Table 9 contains the column names with the highest Chebyshev distance. All distances displayed are > 0.14. Both *"HomeOwner"* and *"MaritalStatus"* are the sampled columns that have the largest distances from the compared columns.

Table 9: Chebyshev distances with weak similarity between Sampled and Compared Column Name using `text-embedding-3-small`

Sampled Column Name	Compare Column Name	Chebyshev Distance
Prefix	HomeOwner	0.149267
HomeOwner	StockDate	0.146625
HomeOwner	FirstName	0.141273
HomeOwner	StockDate	0.140380

Given what we know from Table 8, further examination of the column with the smallest distance could provide further insights. Since *"Prefix"* has the smallest distance, it will be analyzed in greater detail.

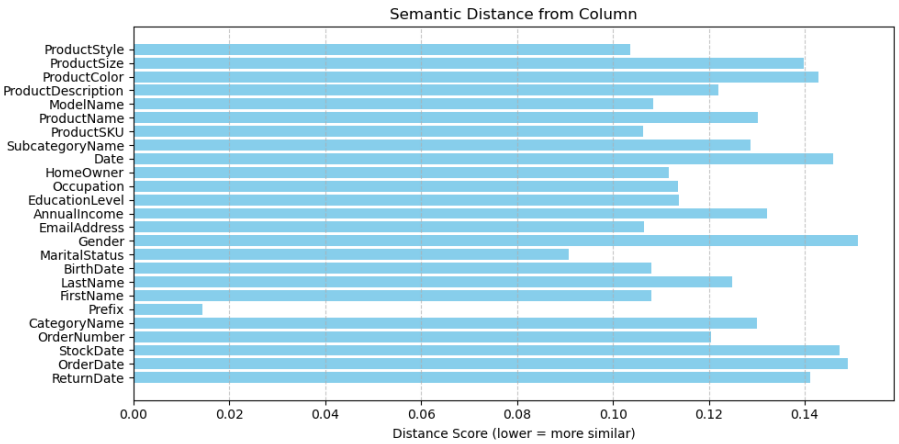

Figure 4: Semantic Distance from column "Prefix"

The analysis shown in Figure 4, examines the distances between the sampled column *"Prefix"* and all the compared columns. *"Prefix"* has the smallest distance with itself. Additionally, *"MaritalStatus"* and *"ProductStyle"* also show a small distance, though not as significant a *"Prefix"*. In contrast, there is a clear separation between *"Prefix"* and *"Gender"*. This indicates a significant semantic separation between them.

5.2 ***Clustering Analysis Results***

To validate the application and the results obtained with the embeddings beyond pairwise similarity distance comparisons, we additionally analyze their structure by reducing the dimension of the embedding space and further implementing a clustering analysis. These techniques will allow us to understand and visualize whether semantically coherent clusters are formed by reducing dimensionality and clustering among various datasets.

To begin, two different dimensionality reduction techniques have been applied: PCA and UMAP. PCA serves as the baseline model for dimensionality reduction, while UMAP is a nonlinear learning algorithm that captures local distances and global structure of the data Healy (2024). Having applied both dimensionality reduction techniques, UMAP was deemed to be the best at conserving data structure while minimizing computational complexity. PCA results can be found in the Appendix B if there is a need for further exploration. Furthermore, three different clustering models have been implemented, the results of which will be evaluated in Section 6.

To determine the best clustering technique for our data, three methods–K-Means, Agglomerative clustering, and HDBSCAN–were tested. The default hyperparameters used to implement UMAP for all three methods are the following: *n_neighbors= 5, min_dist=0.1, n_components=2, metric= "cosine", random_state=42.* Moreover, the default parameters for each model are:

- K-Means:
  - *n_clusters=2*
  - *random_state=42*

- Agglomerative:
  - *n_clusters=2*
  - *metric= "cosine"*
  - *linkage= "average"*

- HDBSCAN:
  - *min_cluster_size=5*
  - *metric= "euclidean"*
  - *cluster_selection_method= "eom"*

It is important to highlight that in HDBSCAN, the euclidean metric on normalized UMAP space approximates cosine, that is the reason behind the selection of this metric. HDBSCAN's ability to handle data with varying densities, as well as sparse data Asyaky and Mandala (2021b) makes it ideal for further testing on this thesis. The combination of UMAP and HDBSCAN allows for enhanced cluster definition and precise grouping, additionally HDBSCAN's biggest advantage is that it does not need predefined number of clusters, and is well-suited for clustering tasks that involve high-dimensional data, it identifies clusters of various densities which is ideal for our analysis where semantic groups may be formed from tighter to more looser groups.

Considering all of the above we have done thorough hyperparameter tuning to analyze how the results vary with the optimal hyperparameters. For the optimiz ation process of the hyperparameters, Grid Search was employed for all clustering methods, where the algorithm systematically evaluates a defined set of parameter combinations to obtain the optimals.

After hyperparameter tuning the resulting optimal parameters are the following:

- K-Means:
  - *n_clusters= 10*
  - *random_state=42*

- Agglomerative:
  - *n_clusters=10*
  - *metric= "cosine"*
  - *linkage= "average"*

- HDBSCAN:
  - *min_cluster_size=5*
  - *metric= "euclidean"*
  - *cluster_selection_method= "eom"*

We have concluded that the optimal hyperparameter values are the following; for defining UMAP: *n_neighbors= 5, min_dist=0.1, n_components=5,* and for HDBSCAN a *min_cluster_size=5.*

5.2.1 Before hyperparameter tuning

Figure 5 presents a comparison between the three different clustering methods after implementing UMAP as a dimensionality reduction technique. The evaluation done is based on the three different clustering metrics: silhouette score, normalized mutual information and purity.

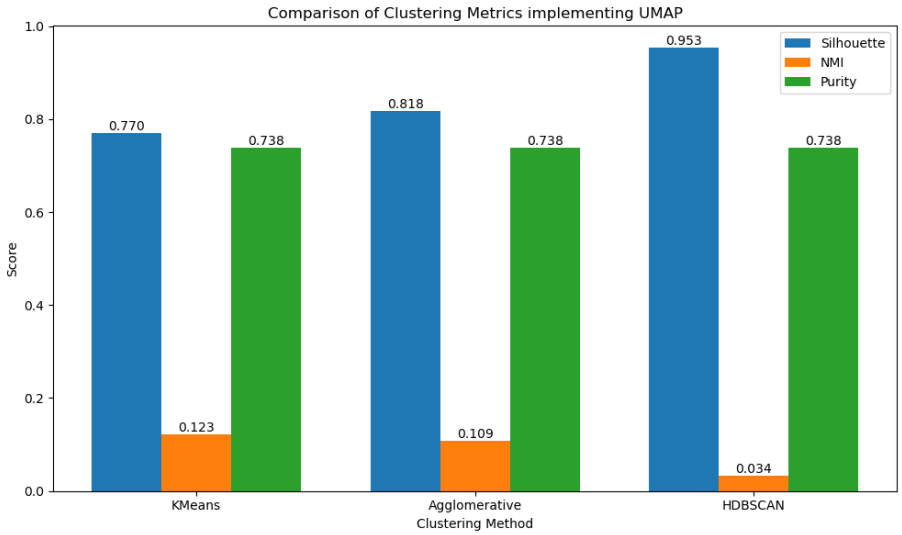

Figure 5: Comparison of Clustering Metrics Implementing UMAP

It can be seen that HDBSCAN achieved the highest silhouette score (0.953), this indicates that it produces the most well-defined clusters, it has the highest intra-cluster cohesion and separation when compared to the other methods, K-Means and Agglomerative clustering. On the contrary, NMI which measures the similarity between predicted and true column categories, has the lowest value for HDBSCAN (0.034). It can be seen it has a lower value for all three methods, this shows that all methods struggle to align clusters with the true categories. Despite this, all three demonstrate to have a moderate purity score (0.738), this metric assesses how much a cluster comprises data points from mainly one single class. Its high score suggests that each cluster tends to contain data points from mostly one class.

Figure 6 shows a 2D UMAP projection of the data where HDBSCAN has been implemented, and identified 3 different clusters.

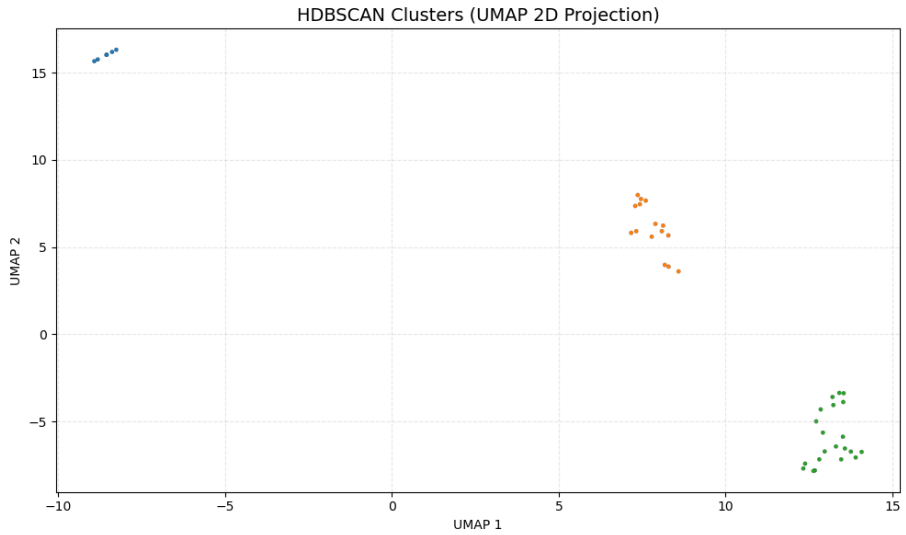

Figure 6: 2D representation of Cluster with default hyperparameters

The three different clusters that can be seen are defined by a total of 42 columns and 0 noise points. Table 10 shows the number of columns that each cluster contains and its names.

Table 10: Cluster Information

Cluster number	**Columns contained**	**Name of columns**
0	6	FirstName, LastName, EmailAddress, FirstName, LastName, EmailAddress
1	15	Date, BirthDate, AnnualIncome, ReturnDate, OrderDate, StockDate, OrderNumber, OrderDate, StockDate, OrderNumber, OrderDate, StockDate, OrderNumber, BirthDate, AnnualIncome
2	21	Prefix, MaritalStatus, Gender, EducationLevel, Occupation, HomeOwner, ProductSKU, ProductName, ModelName, ProductDescription, ProductColor, ProductSize, ProductStyle, CategoryName, SubcategoryName, Prefix, MaritalStatus, Gender, EducationLevel, Occupation, HomeOwner

Cluster 0 is characterized by columns related to *"Personal Identifiers"*, therefore they all follow a similar semantic alignment. Cluster 2 is specially dominated by *"Temporal"* and *"Financial"* data, lastly Cluster 3 is formed by *"Demographic"* and *"Product"* de-

scriptions. This cluster is the largest and most diverse as it includes a mix of categorical data.

5.2.2 After hyperparameter tuning

With the optimal configuration of hyperparameters, HDBSCAN shows the best silhouette score performance out of the three clustering methods (0.962), this indicates that there are better defined-clusters. However, when it comes to NMI (0.047) HDBSCAN underperforms significantly. NMI presumes that K-Means is the best-performing algorithm, while with purity, all three models reflect high dominance of true categories within the clusters. A graphical representation of the following can be seen in Figure 7.

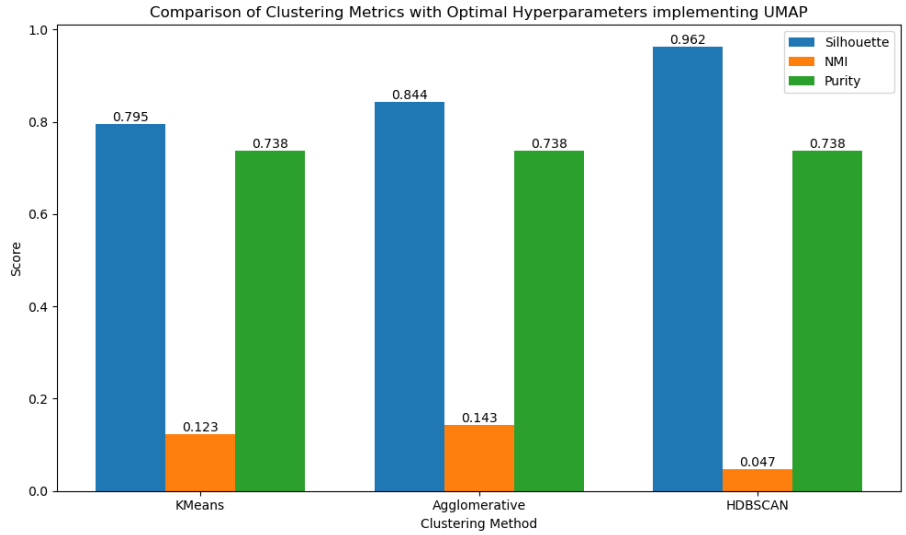

Figure 7: Comparison of Clustering Metrics with Optimal Hyperparameters Implementing UMAP

The ideal configuration of HDBSCAN parameters forms three clusters containing 42 columns in total. The cluster sizes vary significantly, Cluster 0 is made up of 6 columns, Cluster 1 contains 12 columns, and Cluster 2 has 23 columns. Table 11 shows the columns contained in each of the clusters.

Table 11: Cluster Information

Cluster number	Columns contained	Name of columns
0	6	FirstName, LastName, EmailAddress, FirstName, LastName, EmailAddress
1	12	Date, BirthDate, AnnualIncome, ReturnDate, OrderDate, StockDate, OrderDate, StockDate, OrderDate, StockDate, BirthDate, AnnualIncome
2	24	Prefix, MaritalStatus, Gender, EducationLevel, Occupation, HomeOwner, ProductSKU, ProductName, ModelName, ProductDescription, ProductColor, ProductSize, ProductStyle, CategoryName, SubcategoryName, OrderNumber, OrderNumber, OrderNumber, Prefix, MaritalStatus, Gender, EducationLevel, Occupation, HomeOwner

Because the clusters can be analyzed both with the "sampled" and "compare" embeddings, the columns are displayed twice. In one, demonstrating the embeddings of the first half of the dataset while the other represents the second half.

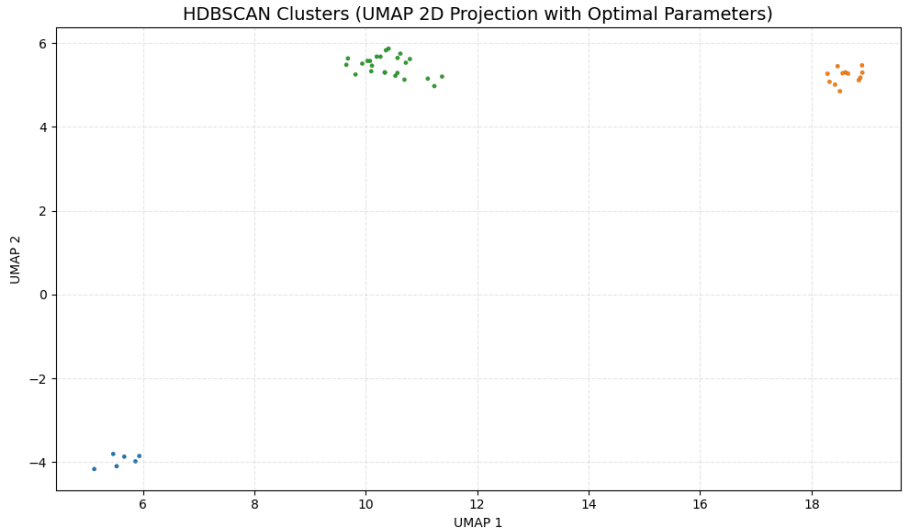

Figure 8: 2D representation of Cluster with optimized hyperparameters

HDBSCAN forms three distinct clusters, as seen in Figure 8. Cluster 0, represented by the color blue, portrays personal identity traits data. Cluster 1, orange cluster cloud, is formed by financial information and dates, meaning all the data in this cluster is numeric. Cluster 2, represented by the color green, is formed by the largest number of columns, and it contains demographic and product information.

6 DISCUSSION

By examining the results, it will be possible to determine if the classification problem successfully identifies similar column names based on their similarity distance. A shorter similarity distance suggests that column names are closely positioned in vector space, while a larger distance indicates that they are located farther apart.

All distance metrics exhibited high accuracy, suggesting all the distance metrics are well-suited for textual data. However, in line with the literature review, "cosine distance stands out as the most efficient metric for textual data" as stated by Gómez and Vázquez (2022). Our findings suggest that embedding textual data preserves the semantic identity of the columns, as shown by the short distances measured on the embeddings when identifying their respective categories. Additionally, this aligns well with clustering analysis, which relies on distance similarity metrics to determine cluster formation. Columns with the largest distances suggest that the content of the clusters differ, potentially due to different casing and/or word boundaries. The format of the data can also contribute to increased distances.

The insights from the clustering analysis provided an overall clear and well-defined representation of the clusters. Prior to hyperparameter tuning, the results obtained indicate that in terms of clustering quality, HDBSCAN is the best-performing model regarding unsupervised performance (silhouette score); it accurately reflects the structure of the data after implementing UMAP.

The lower value of NMI should not be perceived as a weakness, but rather as an indicator of the algorithm's ability to uncover nuanced and significant clusters that may not align with broad categories of the data. Traditional evaluation labels tend to classify columns into larger more generalized categories which might hide nuanced differences between column types. HDBSCAN, on the contrary, focuses on local density differences and this allows to separate broad categories into more specific clusters based on more subtle similarities in their embedding representations. In conclusion, even though the alignment with the categories has a slightly low value, the clustering provides a deeper segmentation of the data.

HDBSCAN does not optimize the number of clusters directly, instead it optimizes the hyperparameter *"min_cluster_size"*. The algorithm focuses on identifying dense regions in the dimensional space to correctly determine the boundaries of each cluster. This justifies why the number of clusters found remained consistent before and after hyperparameter tuning. However, the quality of the clusters improved, especially in terms of separation and density. Additional specific insights regarding the clusters show:

Cluster 0: suggests a strong similarity in the embeddings formed on these columns. Because the data came from both the sampled as well as the compare dataset, the data formation is evenly split between them. Cluster 1: proposes a certain pattern formed on the embeddings of this data which proposes a similar structure on both of these data types. Cluster 2: suggests an underlying relationship between products and demographic client information, marking some correlation between these two categories.

7 ADDITIONAL VALIDATION RESULTS

To complement the primary classification results obtained when using our Database A, and gain further insight into the column classification and clustering analysis we have conducted additional validation on the other 2 Databases presented in the previous section 3, Database B and Database C. These results aim to assess the consistency of the model built to embed text data and analyze the consistency of the classification using semantic similarity distances and different clustering methods. Together these results will provide a deeper understanding of the structure of the model and determine the success of embedded data for column classification.

7.1 ***Database B***

7.1.1 Cosine Distance Results

Results based on the cosine distance metric shown in Table A1 (See Appendix), are the lowest distances found. The model successfully captures semantic identity of the columns, the embeddings of these column names are nearly identical, therefore it reflects a very strong classification when matching column names, which is expected. On the other hand, Table A2 (See Appendix) shows the highest-distance pairs, with the poorest matching pairs or ambiguous semantic relationships. Although the distances are quite large, the largest one has a value of 0.6916 which corresponds to *"Category"* and *"Product Name"*, this pair for example might be conceptually related in some domains, but they are not necessarily similar in wording, that is the reason why the model can misclassify them. Other pairs, as *"Currency"* and *"Date"* are indeed quite different, then the high distance value is expected to be.

7.1.2 Clustering Results

Pre-tuning

To determine the best clustering method for our data, as stated previously the same clustering methods will be implemented. The default hyperparameters used to implement UMAP for all three methods are the following: *n_neighbors= 5, min_dist=0.1, n_components=2, metric= "cosine", random_state=42.* Moreover, the default parameters for each model are:

- K-Means:
  - *n_clusters=2*
  - *random_state=42*

- Agglomerative:
  - *n_clusters=2*
  - *metric= "cosine"*
  - *linkage= "average"*

- HDBSCAN:
  - *min_cluster_size=5*
  - *metric= "euclidean"*
  - *cluster_selection_method= "eom"*

Figure B3 (See Appendix), shows the results for the clustering analysis after implementing UMAP for dimensionality reduction. Based on the metrics, Agglomerative clustering achieves the highest silhouette score (0.720), successfully separating the clusters and obtaining high cohesion. All three methods achieve the same purity score (0.769), this suggests that they all yield the same ability to group semantically coherent columns. Finally, HDBSCAN slightly outperforms the others in NMI, but by a very minimal difference. In conclusion, we would argue that for Database B, the best performing clustering model is Agglomerative.

Figure B2 (See Appendix) shows a 2D representation of the cluster structure analysis. The UMAP projection of Agglomerative clustering shows 2 clear clusters. Cluster 0 with 48 columns, and Cluster 1 with 30 columns. Table A3 (See Appendix) shows in detail the columns that form each cluster. Cluster 0 contains mostly structural, demographic and identifier-type columns, like: *"Order ID", "Date", "Size",* etc. On the other hand, Cluster 1, includes more transactional, operational column types, some examples are: *"Status", "Fulfilment", "Sales Channel"*, etc. The semantic separation is meaningful; Cluster 0 groups identifying columns while Cluster 1 captures more categorical columns. There is an overlap in some like *"Category", "Status", "Date"*, etc. that appear in both clusters, this suggests that those terms might benefit from context-aware embeddings.

In conclusion, Agglomerative clustering with UMAP shows a very strong structure in the embeddings space and clustering analysis, balancing both cohesion and semantic grouping for Database B. The cluster contents demonstrate that the model can successfully distinguish between types of columns (identifier attributes and categorical). Moreover, the results after tuning the hyperparameters will be analyzed.

Post-tuning

For the optimization process of the hyperparameters, Grid Search was employed for all clustering methods. After hyperparameter tuning the resulting optimal parameters are the following:

- K-Means:
  - *n_clusters= 10*

- *random_state=42*

- Agglomerative:
  - *n_clusters=10*
  - *metric= "cosine"*
  - *linkage= "average"*

- HDBSCAN:
  - *min_cluster_size=10*
  - *metric= "euclidean"*
  - *cluster_selection_method= "eom"*

We have concluded that the optimal hyperparameter values are the following: for defining UMAP: *n_neighbors=5, min_dist=0.5, n_components=5*, and for Agglomerative: *n_clusters=10*.

Figure B3 (See Appendix) shows how the same clustering metrics have improved after tuning the hyperparameters. It is visible that for Agglomerative clustering the silhouette score increased dramatically (0.824), while the NMI dropped slightly (0.043) it is acceptable given that there is a better structural separation within the clusters. Purity remained constant (0.769), again for all three methods. Additionally, K-Means has benefited from tuning, specially with the increase of the silhouette score (0.272), however it is still outperformed by the other 2 methods. Finally, HDBSCAN has had a very significant increase, this model shows the largest relative percentage increase, especially in silhouette score, even if its absolute performance remains below Agglomerative clustering.

Figure B4 (See Appendix) shows the 2D representation of the clusters after tuning the hyperparameters and Table A4 (See Appendix) shows the composition of each cluster. It can be seen that after tuning the hyperparameters the model has definitely improved in cohesion and granularity as the custers now reflect more semantically tight groups.

In conclusion, Agglomerative clustering with optimal hyperparameters is the best performing method for Database B, based on quantitative metrics and qualitative cluster content. Post-tuning results show a very much well improved cluster segmentation and coherent groupings with meaningful semantic patterns, this validates the strength of the embeddings and the clustering approach.

7.2 ***Database C***

7.2.1 Cosine Distance Results

Cosine similarity was used to determine the classification of columns from Database C. Table A5, provides the new column names whose distances are the smallest out of all the

distances found. The smallest distance is between *"CalendarHalfYearLabel"* and *"FiscalHalfYearLabel"*, which has a value of 0.006815. Although the column is incorrectly classified it is important to address how both columns contain the same information, a year. Because the data is indistinguishable, the classification of *"CalendarHalfYearLabel"* may appear as *"FiscalHalfYearLabel"*.

Table A6, represents the largest distances found for Database C. The bigger the distance between the new column name and the old column name, the bigger the sematic difference between them will be.

"CalendarDayOfWeekLabel" and *"StoreType"* have the highest distance in this database (See in table A6), they have a distance >0.62 units. Because of the large distance, it is assumed that there are disparities in the data contained in these columns, hence the distances between their location in vector space are greater. *"CalendarDayOfWeekLabel"* contains the days of the week, while *"StoreType"* contains the description of the store if they are characterized for being stores, catalogs, resellers, or online.

Because *"CalendarHalfYearLabel"* contains the smallest distance from all the embeddings, this column was selected to further analyze its behavior with all of the columns contained in Database C.

As explained previously, the indistinguishable differences between the content of the columns *"CalendarHalfYearLabel"* and *"FiscalHalfYearLabel"* is reflected in Figure B5. Both of these columns have a lowest distance between them entailing a similarity in the data. Although this can be seen as an error, this is reflective of the model's ability to correctly classify columns that contain the same or similar information.

Although this may be vaguely called a misclassification, this error is rectified by determining the correct column classifications as a close distance as well. These findings validate the use of similarity metrics to uncovering columns with similar data as well as its use for classification purposes.

7.2.2 Clustering Results

Pre-tuning

The default hyperparameters used to implement UMAP for all three methods are the following: *n_neighbors= 5, min_dist=0.1, n_components=2, metric= "cosine", random_state=42.* Moreover, the default parameters for each model are:

- K-Means:
  - *n_clusters=2*
  - *random_state=42*
- Agglomerative:
  - *n_clusters=2*

  - *metric= "cosine"*
  - *linkage= "average"*

- HDBSCAN:
  - *min_cluster_size=5*
  - *metric= "euclidean"*
  - *cluster_selection_method= "eom"*

Figure B6 represents the performance metrics of the three models. It can be observed how the best Silhouette score is achieved with Agglomerative Clustering, portraying how with this method, the columns depicted in the clusters are compact and the clusters are separated from each other. The high Purity score is the same score for all the clusters, it depicts how the categories of the columns inside of the clusters are correctly matching with each other. Agglomerative clustering also has the highest NMI, demonstrating how the labels in the clusters are preserved.

Agglomerative clustering technique with default hyperparameters generates 2 clusters and 0 noise points. Cluster 0 contains the majority of the columns with a total of 130 columns. Cluster 1 contains 4 columns representing a better defined cluster than Cluster 0. The description of the content of each cluster can be seen in Table A7. Figure B7 is a visual representation of the cluster formation. Cluster 0 is represented by the blue color and Cluster 1 by the orange color.

Post-tuning

For the optimization process of the hyperparameters, Grid Search was employed for all clustering methods. After hyperparameter tuning the resulting optimal parameters are the following:

- K-Means:
  - *n_clusters= 10*
  - *random_state=42*

- Agglomerative:
  - *n_clusters=10*
  - *metric= "cosine"*
  - *linkage= "average"*

- HDBSCAN:
  - *min_cluster_size=10*
  - *metric= "euclidean"*

– *cluster_selection_method= "eom"*

We have concluded that the optimal hyperparameter values are the following: for defining UMAP: *n_neighbors=5, min_dist=0.5, n_components=5*, and for Agglomerative: *n_clusters=10*.

Agglomerative Clustering is selected as the best clustering method because of the high silhouette score, prompting an optimization of this technique to establish the optimal cluster.

With the optimal hyperparameters new results from the performance evaluation metrics can be seen in Figure B8. Silhouette score now increases to 0.834. Yet when it comes to NMI, the performance decreases to a score of 0.031. Purity remains the same for all of the clustering techniques. Agglomerative clustering forms 10 clusters and 0 noise points each of them encompassing a set of columns from Database C. The columns contained in the cluster range from 4 to 34. Table A8 shows the clusters, the amount of columns per cluster and the names of the columns in the cluster.

In order to understand the distribution of the clusters, Figure B9 shows all clusters are defined, however there is some overlapping, even with the "optimal" hyperparameters, that reinforces that some points between clusters might have ambiguous assignments. Overall, the plot shows distinct clusters and minimal noise.

8 IMPACT OF DATASET TRAITS ON CLUSTERING

In this thesis, the three databases—Database A, B, and C— as well as their respective datasets, have distinct structures and characteristics. Their differences allows the methodology to be tested across different data types. Although the methodology ideally creates a universal method with which the classification and clustering of columns is accurate, we acknowledge the limitations that this technique may have and the tuning that might be needed for other databases.

An interesting aspect that has been observed is the variation of the best clustering algorithms depending on the dataset that is chosen as the "compare" set. To better explain how the data type impacts the clustering results, this section will be divided into two parts. The first division will describe the key characteristics that make each of the databases different. The second division will highlight how each of the clustering methods performs better with a dataset that fulfills certain characteristics.

8.1 ***Dataset traits***

The data structure of each database plays a crucial role in shaping outcomes. Database A contains certain numerical columns—"AnnualIncome," "Date", "ReturnData", "OrderDate", "StockDate", "OrderNumber," and "BirthDate"—that may impact the results found. Database B and C also contain certain numerical columns, however, they are composed of a higher amount of textual columns compared to numeric. Table 12 displays the amount of textual columns, numerical columns, as well as the percentage describing the proportion of numerical data.

Table 12: Overview of database structures, including text, numerical columns, and the percentage of numerical data.

Databases	**Text Columns**	**Numeric Columns**	**Percentage of Numerical Data**
Database A	23	10	30.3%
Database B	51	7	12.1%
Database C	89	28	23.9%

Each database with which we are working exhibits different characteristics, making the data increase in complexity. Database A contains interconnected datasets, accounting for the possibility of relationships between these datasets, yet it has the least amount of columns and the most numerical data.

Database B works with independent datasets, this accounting for fewer relationships between the datasets. However, this database contains the least amount of numerical data, aiding in understanding the performance of clusters in a dataset that is mostly composed of textual data. The complexity of the data is mediated by having less data than Database C yet more than Database A, as well as the independence of the datasets.

Database C contains the most data out of all the databases, equating to a higher computational complexity as well as requiring a clustering model that is able to handle high complexity. The combination of the datasets' interconnectivity and the high dimensionality of the data increases complexity.

8.2 ***Clusters and method selection***

Each of the clustering techniques analyzed in this thesis has its own unique way of identifying clusters, as described in Section 4. Clustering techniques are highly dependent on the similarities found in the data (Petukhova et al., 2025). Knowing this, the model's ability to cluster data together, also derives from the complexity of said data, as does the quality of its interpretations (Petukhova et al., 2025).

Although the results demonstrated that all the techniques were able to form well-defined clusters, the best-defined and formed clusters are the ones that best interpret the data. In this case, we detected how different cluster techniques work best for data with certain characteristics by measuring three separate performance metrics of the clusters—Silhouette score, NMI, and Purity.

8.2.1 K-Means clustering

Although this algorithm is one of the most popular, the performance of K-Means is best seen only in numerical data, defining the model's strength as such. This occurs due to the need to find distance metrics between the data points, which rely on numerical data (Ahmed et al., 2020). When conducting clustering, selecting features that correctly represent the data is an important step (Jain, 2010). If the features selected are good, the clusters will be isolated and compact (Jain, 2010). Because the databases have been applied to the clustering algorithms without having a preliminary feature selection, the algorithms may determine certain columns to be irrelevant or noise and base the cluster in relation to those columns rather than finding actual relationships.

Couto (2005) evaluates the use of K-Means on embedded categorical data, which allowed for a deeper exploration of the relationship between the data. She discovered that by embedding what was originally categorical data, the clusters were preserved. Since the databases we are working with have been embedded, K-Means may help find unseen patterns in the data.

8.2.2 Agglomerative Clustering

When it comes to Database B—primarily containing textual data with an independence of its datasets—agglomerative clustering is found to have the best performance due to its ability to cluster data that lies close in vectoral space (Dillon and Goldstein (1984), as cited in Oti and Olusola (2024b)). The unknown number of clusters gives this method the freedom to create clusters, according to the linkage prerequisite, until the data is fully

formed into one cluster. This allows for the partition to be set to the best-fitting amount of clusters (Emmendorfer and de Paula Canuto, 2021).

Although the complexity in Database C increases, agglomerative clustering still performs the best. The hierarchical nature of agglomerative clustering may help in creating the clusters for the database. Due to the interconnectivity of the data, the similarities between certain data points cause the merging of the clusters (Oti and Olusola, 2024a).

The data set chosen for the comparison also dictates the structure the clusters will have. If the data chosen for comparison is hierarchical in nature, the best clustering technique will tend to be agglomerative due to the interrelationships that exist between the columns of the data (Tokuda et al., 2022). They state how the tuning parameters and the methodology selected to analyze the dataset are important to the performance of the clustering technique. The results of the model vary according to these parameters.

8.2.3 HDBSCAN

As stated previously, HDBSCAN is the best-performing model for Database A, with respect to its unsupervised performance (silhouette score). Its powerful density-based focus allows it to separate broad categories into more specific clusters; producing hierarchical clustering Campello et al. (2013). The quality of this technique is dependent on some key aspects. It is very sensitive to the choice of text embeddings; contextual embeddings can significantly outperform non-contextual or more traditional embeddings, Saha (2023a) argues. Additionally, the combination with UMAP for dimensionality reduction shows how it improves its efficiency, especially by reducing runtime when working with large datasets and higher embedding dimensions.

9 CONCLUSION

In this thesis we have examined and performed a comparison on various distance similarity metrics and clustering algorithm to reveal if the use of embeddings on text data could allow the correct classification of column names.

Our results, based on the primary Database A (stated by Adage) have demonstrated that overall all distance metrics analyzed: Cosine distance, Euclidean distance, Manhattan distance and Chebyshev distance, achieve a high accuracy. This indicates their applicability on embedded textual data for column classification. It is important to highlight that Cosine distance can be regarded as the most effective metric when dealing with high-dimensional data, in line with the findings on literature of previous study done. Considering the above, we can confirm that the use of embeddings for textual data not only preserved the semantic identity of the columns but also enhanced the ability to identify column names, facilitating the classification process.

Regarding the clustering analysis, the combination of UMAP for dimensionality reduction and HDBSCAN, provided a clear representation of the clusters with the silhouette score (silhouette = 0.949) outperforming all other metrics. The choice of the embeddings, in this case static (text-embedding-3-small model), proper hyperparameter tuning (using Grid Search and cosine metric) and its ability to identify outliers or noise embeddings, consolidate why this method has been the most successful.

Despite offering valuable insights, our thesis has some limitations. The similarity distances may have been impacted by various aspects of the text data, some of which are: different casing, data format or word boundaries. To obtain the most robust and accurate results possible, we worked with several datasets, as previously discussed. Our results based on Database A are satisfactory with these methods. However, we do not rule out that using other types of data or formats could alter the results due to how the models and distance measurements adapt to them.

The generalizability of our results can be constrained by the datasets used. In fact, it can be seen in Section 7 that the best-performing clustering method may vary depending on the data used, nonetheless, it should be highlighted that the technique for dimensionality reduction UMAP, remains as the most satisfactory, due to the use of high-dimensional data. In that regard, future research should aim to address these limitations by exploring even more diverse datasets and refining the methods utilized.

Our thesis exemplifies the power of combining new methodologies with domain-specific expertise to tackle common traditional challenges in the field of data management and analysis. The results obtained and conclusion can have influence in more research and practical approaches in both academic and professional and practical contexts. We have successfully proven the use of methods for textual data embedding for column classification and clustering.

10 FUTURE RESEARCH

As mentioned previously, the methodology used to address the research questions in this thesis presents a new solution to a problem that companies continuously experience. This application can serve as a foundation for the use of embeddings in databases for text data classification, allowing for meaningful contributions to this field while preventing redundant research.

The research for this thesis was conducted using Database A and validated with Databases B and C. However, as discussed previously, due to the variability of text data, results may differ in different settings. Additionally, testing this methodology on various types of data may provide a more precise determination of the accuracy of the distance similarity measures used. The practical application of this approach in a corporate setting must be adapted to the data provided by the respective organization.

This analysis did not incorporate embeddings for numerical features found on the databases due to a focus on textual data. Previous research have generated embeddings on numerical features to improve model performance, however the research has mostly been conducted on deep learning. For example, Gorishniy et al. (2023) explored embeddings on tabular data, and Wallace et al. (2019) successfully applied embedding to numerical data. Although this investigation focuses on textual data, expanding the methodology to include numerical data could add depth to the analysis.

Exploring alternative distance metrics can help determine which methods correctly represent data for classification purposes. Most distance similarity metrics are best achieved in a certain type of text. The combination of various distance metrics is a methodology that may also accentuate the findings that have been made. For instance, Rodrigues (2021) discussed the combination of Minkowski and Chebyshev distances to improve classification accuracy. This combination may allow for multiple text types to be analysed.

Clustering methods aid in understanding the formation of the data in vector space, and this investigation analyses three different clustering techniques. Further research may explore diverse clustering techniques that can handle the high variability that occurs with textual data.

In conclusion, while this analysis creates a solid base for the investigation of embeddings on databases, further research is encouraged. There are many avenues to explore, adaptation of the methodology onto new databases, the addition of numerical data to the embeddings, alternative distance metrics or even the combination of such, and the exploration of other clustering techniques.

References

Abdallah, I. (2024). Dk housing prices sample 100k dataset.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In Van den Bussche, J. and Vianu, V., editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ahmed, M., Seraj, R., and Islam, S. M. S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295.

Alvarez, J. E. and Bast, H. (2017). A review of word embedding and document similarity algorithms applied to academic text. *Bachelor thesis*, 1.

Arpit2712 (2024). Amazon sales report dataset.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–9. Society for Industrial and Applied Mathematics.

Asyaky, M. S. and Mandala, R. (2021a). Improving the performance of hdbscan on short text clustering by using word embedding and umap. In *2021 8th international conference on advanced informatics: Concepts, theory and applications (ICAICTA)*, pages 1–6. IEEE.

Asyaky, M. S. and Mandala, R. (2021b). Improving the performance of hdbscan on short text clustering by using word embedding and umap. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–6.

Bansal, L. (2024). Sales of a supermarket dataset.

Bhadra, M. (2024). World bank dataset.

Campello, Ricardo J. G. B., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., and Xu, G., editors, *Advances in Knowledge Discovery and Data Mining – PAKDD 2013*, volume 7819 of *Lecture Notes in Computer Science*, pages 160–172, Berlin, Heidelberg. Springer.

Couto, J. (2005). Kernel k-means for categorical data. In *International symposium on intelligent data analysis*, pages 46–56. Springer.

da Costa, L. S., Oliveira, I. L., and Fileto, R. (2023). Text classification using embeddings: a survey. *Knowledge and Information Systems*, 65(7):2761–2803.

Dehghani, Z. (2020). How to move beyond a monolithic data lake to a distributed data mesh. Accessed: 2025-03-12.

Dillon, W. R. and Goldstein, M. (1984). *Multivariate analysis: Methods and applications*. New York (NY): Wiley, 1984.

Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., and Tang, N. (2017). Deeper: Deep entity resolution. *arXiv preprint arXiv:1710.00597*.

Emmendorfer, L. R. and de Paula Canuto, A. M. (2021). A generalized average linkage criterion for hierarchical agglomerative clustering. *Applied Soft Computing*, 100:106990.

Forster, J. (2024). Federated learning and data mesh: how it enhances data architecture. Accessed: 2025-03-14.

Gibin, W. O. (2024). Customer churn dataset.

Gómez, J. and Vázquez, P.-P. (2022). An empirical evaluation of document embeddings and similarity metrics for scientific articles. *Applied Sciences*, 12(11):5664.

Goodman-Bacon, A. (2021). Difference-in-differences with variation in treatment timing. *Journal of econometrics*, 225(2):254–277.

Gorishniy, Y., Rubachev, I., and Babenko, A. (2023). On embeddings for numerical features in tabular deep learning.

Harris, Z. S. (1954). Distributional structure. *WORD*, 10(2-3):146–162.

Healy, J., M. L. (2024). Uniform manifold approximation and projection. *Nat Rev Methods Primers*, 4.

Hou, Y., Li, J., Fang, Z., and Zhang, X. (2020). An initialization method of deep q-network for learning acceleration of robotic grasp. In *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6. IEEE.

Hulsebos, M., Hu, K., Bakker, M., Zgraggen, E., Satyanarayan, A., Kraska, T., Demiralp, Ç., and Hidalgo, C. (2019). Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD International Conference on knowledge discovery & data mining*, pages 1500–1508.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

Jin, P., Zhang, Y., Chen, X., and Xia, Y. (2016). Bag-of-embeddings for text classification. In *IJCAI*, volume 16, pages 2824–2830.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, 2 edition.

Li, H. and Toor, S. (2024). Empowering data mesh with federated learning. In *2024 IEEE International Conference on Big Data (BigData)*, pages 2340–2342.

Li, Saihan and Gong, Bing (2021). Word embedding and text classification based on deep learning methods. *MATEC Web Conf.*, 336:06022.

Malaiarasugraj (2024). E-commerce dataset.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). Vector space classification. In *Introduction to Information Retrieval*, chapter 14, pages 290–325. Cambridge University Press, Cambridge, UK. Accessed: 2025-05-15.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Michel Marie Deza, E. D. (2013). *Encyclopedia of Distances*. Springer, Paris, Moscow, 2 edition.

Microsoft (2025). Adventureworks database sample.

Mitri, M. (2015). Active learning via a sample database: the case of microsoft's adventure works. *Journal of Information Systems Education*, 26(3):177–186.

Murtagh, F. and Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of classification*, 31:274–295.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Oti, E. and Olusola, M. (2024a). Overview of agglomerative hierarchical clustering methods. *British Journal of Computer, Networking and Information Technology*, 7:14–23.

Oti, E. U. and Olusola, M. O. (2024b). Comparative evaluation of six agglomerative hierarchical clustering methods with a robust example. *African Journal of Mathematics and Statistics Studies*, 7(2):1–25.

Patel, S. (2024). Bank additional full dataset.

Petukhova, A., Matos-Carvalho, J. P., and Fachada, N. (2025). Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, 6:100–108.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

Reimers, N., Schiller, B., Beck, T., Daxenberger, J., Stab, C., and Gurevych, I. (2019). Classification and clustering of arguments with contextualized word embeddings. *arXiv preprint arXiv:1906.09821*.

Rodrigues, E. O. (2021). Combining minkowski and chebyshev: New distance proposal and survey of distance metrics using k-nearest neighbours classifier. *CoRR*, abs/2112.12549.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Saha, R. (2023a). Influence of various text embeddings on clustering performance in nlp. *arXiv preprint arXiv:2305.03144*.

Saha, R. (2023b). Influence of various text embeddings on clustering performance in nlp. *arXiv preprint*. Accessed:19/05/2025.

Shvaiko, P. and Euzenat, J. (2013). Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176.

Steck, H., Ekanadham, C., and Kallus, N. (2024). Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, pages 887–890.

Thakur, B. (2023). Cleaned contoso dataset.

Tokuda, E. K., Comin, C. H., and Costa, L. d. F. (2022). Revisiting agglomerative clustering. *Physica A: Statistical mechanics and its applications*, 585:126433.

Upadhye, A. (2023a). Enhancing semantic understanding by visualizing sentence-level embeddings. *International Journal of Computer Applications*, 185(46):1–5.

Upadhye, A. (2023b). Enhancing semantic understanding by visualizing sentence-level embeddings with umap. *International Journal of Computer Applications*, 185(46).

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. (2019). Do nlp models know numbers? probing numeracy in embeddings.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14:1–37.

Appendix A: Tables

Table A1: Smallest cosine distances for column names, Database B

Sampled Column Name	Compared Column Name	Distance
Courier Status	Courier Status	0.016509
Fulfilment	Fulfilment	0.009994
Sales Channel	Sales Channel	0.000001
Status	Status	0.025279

Table A2: Biggest cosine distances for column names, Database B

Sampled Column Name	Compared Column Name	Distance
Category	Product Name	0.691647
Category	Style	0.665040
Category	Sales_type	0.661953
Currency	Date	0.616923

Table A3: Columns in each cluster before optimization, Database B

Cluster Number	Columns Contained	Name of Columns
0	48	Order ID, Date, Style, SKU, Category, Size, ASIN, ship-city, ship-state, job, marital, education, contact, month, day_of_week, Surname, Geography, Gender, date, quarter, house_type, sales_type, address, city, area, region, Product ID, Supplier ID, Customer Age Group, Customer Location, Customer Gender, Invoice ID, Branch, City, Customer type, Gender, Date, Time, Country, Order ID, Date, Style, SKU, Category, Size, ASIN, ship-city, ship-state
1	30	Status, Fulfilment, Sales Channel, ship-service-level, Courier Status, currency, ship-country, promotion-ids, fulfilled-by, default, housing, loan, poutcome, y, Card Type, Product Name, Category, Shipping Method, Seasonality, Product line, Payment, Status, Fulfilment, Sales Channel, ship-service-level, Courier Status, currency, ship-country, promotion-ids, fulfilled-by

Table A4: Columns in each cluster after optimization, Database B

Cluster Number	Columns Contained	Name of Columns
0	3	Geography, Customer Location, Country
1	18	Fulfilment, Sales Channel, ship-service-level, currency, ship-country, promotion-ids, Card Type, Product Name, Category, Shipping Method, Product line, Payment, Fulfilment, Sales Channel, ship-service-level, currency, ship-country, promotion-ids
2	12	Size, ASIN, job, education, Surname, Gender, Customer Gender, Branch, Customer type, Gender, Size, ASIN
3	5	month, day_of_week, Customer Age Group, Time, Date
4	11	SKU, Category, default, housing, loan, poutcome, y, sales_type, Seasonality, SKU, Category
5	10	Order ID, Date, contact, date, quarter, Product ID, Supplier ID, Invoice ID, Date, Order ID
6	5	ship-city, ship-state, City, ship-city, ship-state
7	4	address, city, area, region
8	6	Status, Courier Status, fulfilled-by, Status, Courier Status, fulfilled-by
9	4	Style, marital, house_type, Style

Table A5: Smallest cosine distances for column names, Database C

Sampled Column Name	Compared Old Column Name	Cosine Distance
CalendarHalfYearLabel	FiscalHalfYearLabel	0.006815
CalendarHalfYearLabel	CalendarHalfYearLabel	0.006860
CalendarQuarterLabel	CalendarQuarterLabel	0.006959
FiscalHalfYearLabel	FiscalHalfYearLabel	0.006833

Table A6: Biggest cosine distances for column names, Database C

Sampled Column Name	Compared Old Column Name	Cosine Distance
CalendarDayOfWeekLabel	StoreType	0.628627
CalendarDayOfWeekLabel	AsiaSeason	0.624268
FiscalMonthLabel	DateKey	0.569170
FiscalMonthLabel	CalendarYearLabel	0.565995

Table A7: Columns in each cluster before optimization, Database C

Cluster Number	**Columns Contained**	**Name of Columns**
0	130	SalesTerritoryName, SalesTerritoryRegion, SalesTerritoryCountry, SalesTerritoryGroup, Status, PromotionName, PromotionType, PromotionCategory, DateKey, DateKey, CostType, AccountName, AccountDescription, AccountType, ValueType, DateKey, OutageStartTime, OutageEndTime, DateKey, DateKey, CalendarYearLabel, CalendarHalfYearLabel, CalendarMonthLabel, CalendarWeekLabel, CalendarDayOfWeekLabel, FiscalYearLabel, FiscalHalfYearLabel, FiscalMonthLabel, IsWorkDay, IsHoliday, EuropeSeason, NorthAmericaSeason, AsiaSeason, ProductCategoryName, DateKey, FirstName, LastName, BirthDate, MaritalStatus, Gender, Education, Occupation, ParentEntityLabel, EntityName, EntityDescription, EntityType, StartDate, Status, DateKey, StoreType, StoreName, Status, OpenDate, CloseDate, StorePhone, StoreFax, CloseReason, LastRemodelDate, ChannelName, ChannelDescription, OutageName, OutageType, OutageTypeDescription, OutageSubType, ProductSubcategoryName, DateKey, DateKey, SalesOrderNumber, CurrencyName, CurrencyDescription, ProductName, ProductDescription, Manufacturer, BrandName, ClassName, StyleName, ColorName, WeightUnitMeasureID, UnitOfMeasureName, StockTypeName, AvailableForSaleDate, Status, FirstName, LastName, Title, HireDate, BirthDate, EmailAddress, Phone, EmergencyContactName, EmergencyContactPhone, Gender, DepartmentName, StartDate, Status, SalaryStatus, IsSalesPerson, IsMarried, ScenarioName, MachineType, MachineName, MachineDescription, VendorName, MachineOS, MachineSource, MachineHardware, MachineSoftware, Status, ServiceStartDate, DecommissionDate, LastModifiedDate, GeographyType, ContinentName, CityName, StateProvinceName, RegionCountryName, DateKey, CalendarYearLabel, CalendarHalfYearLabel, CalendarMonthLabel, CalendarWeekLabel, CalendarDayOfWeekLabel, FiscalYearLabel, FiscalHalfYearLabel, FiscalMonthLabel, IsWorkDay, IsHoliday, EuropeSeason, NorthAmericaSeason, AsiaSeason
1	4	CalendarQuarterLabel, FiscalQuarterLabel, CalendarQuarterLabel, FiscalQuarterLabel

Table A8: Columns in each cluster after optimization, Database C

Cluster Number	Columns Contained	Name of Columns
0	21	Status, PromotionName, PromotionType, IsHoliday, EuropeSeason, NorthAmericaSeason, AsiaSeason, Education, Status, Status, ClassName, StockTypeName, Status, Status, IsSalesPerson, IsMarried, Status, IsHoliday, EuropeSeason, NorthAmericaSeason, AsiaSeason
1	14	ProductCategoryName, OutageName, OutageType, OutageTypeDescription, OutageSubType, ProductSubcategoryName, ProductName, ProductDescription, ColorName, MachineName, MachineDescription, MachineOS, MachineHardware, MachineSoftware
2	34	DateKey, DateKey, DateKey, OutageStartTime, OutageEndTime, DateKey, DateKey, CalendarYearLabel, FiscalYearLabel, DateKey, BirthDate, StartDate, DateKey, OpenDate, CloseDate, LastRemodelDate, DateKey, DateKey, SalesOrderNumber, StyleName, WeightUnitMeasureID, UnitOfMeasureName, AvailableForSaleDate, HireDate, BirthDate, StartDate, ScenarioName, MachineType, ServiceStartDate, DecommissionDate, LastModifiedDate, DateKey, CalendarYearLabel, FiscalYearLabel
3	21	SalesTerritoryName, SalesTerritoryRegion, SalesTerritoryCountry, SalesTerritoryGroup, PromotionCategory, ParentEntityLabel, EntityType, StoreType, CloseReason, ChannelName, ChannelDescription, CurrencyName, CurrencyDescription, Title, SalaryStatus, MachineSource, GeographyType, ContinentName, CityName, StateProvinceName, RegionCountryName
4	12	FirstName, LastName, EntityName, EntityDescription, StoreName, Manufacturer, BrandName, FirstName, LastName, EmailAddress, EmergencyContactName, VendorName
5	4	CalendarHalfYearLabel, FiscalHalfYearLabel, CalendarHalfYearLabel, FiscalHalfYearLabel
6	13	CalendarMonthLabel, CalendarWeekLabel, CalendarDayOfWeekLabel, FiscalMonthLabel, IsWorkDay, MaritalStatus, Gender, Gender, CalendarMonthLabel, CalendarWeekLabel, CalendarDayOfWeekLabel, FiscalMonthLabel, IsWorkDay
7	4	CalendarQuarterLabel, FiscalQuarterLabel, CalendarQuarterLabel, FiscalQuarterLabel
8	7	CostType, AccountName, AccountDescription, AccountType, ValueType, Occupation, DepartmentName
9	4	StorePhone, StoreFax, Phone, EmergencyContactPhone

Appendix B: Figures

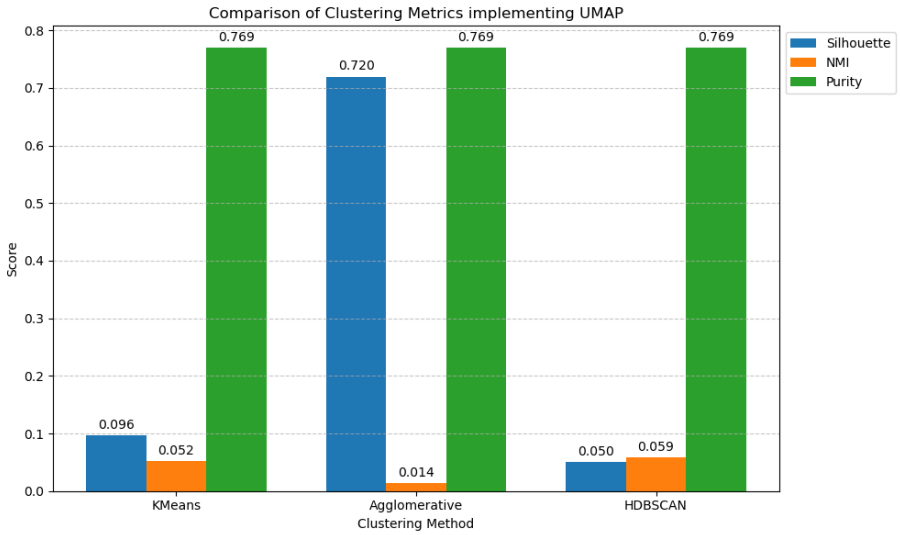

Figure B1: Comparison of Clustering Metrics Implementing UMAP before optimization, Database B

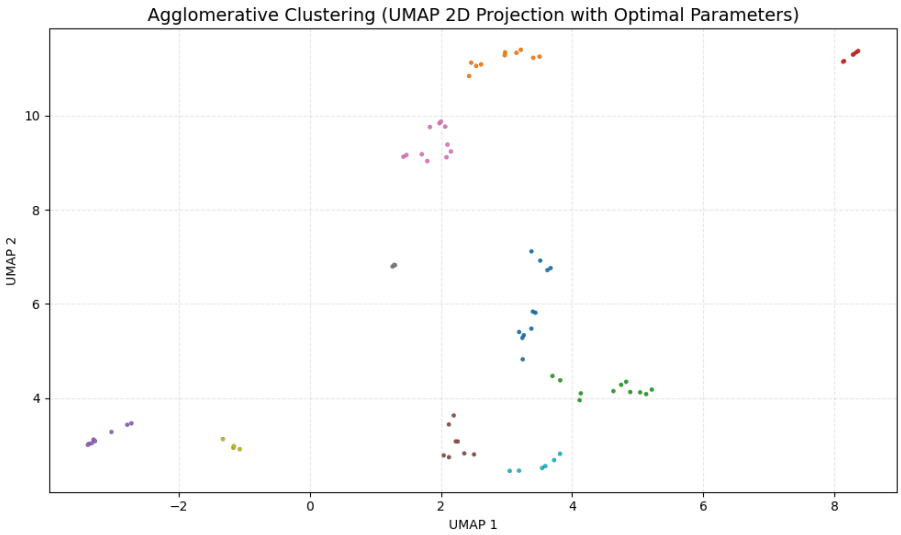

Figure B2: 2D representation of Clusters with original hyperparameters, Database B

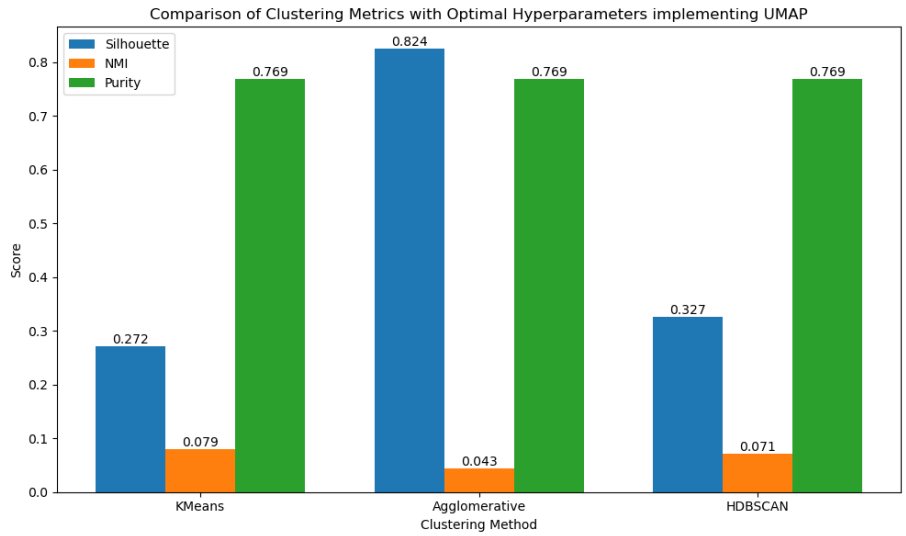

Figure B3: Comparison of Clustering Metrics Implementing UMAP after optimization, Database B

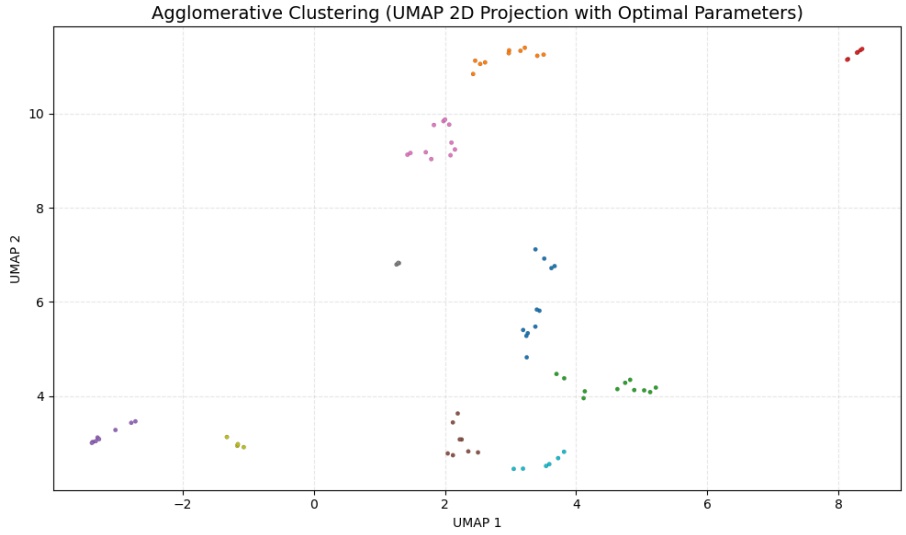

Figure B4: 2D representation of Clusters with optimized hyperparameters, Database B

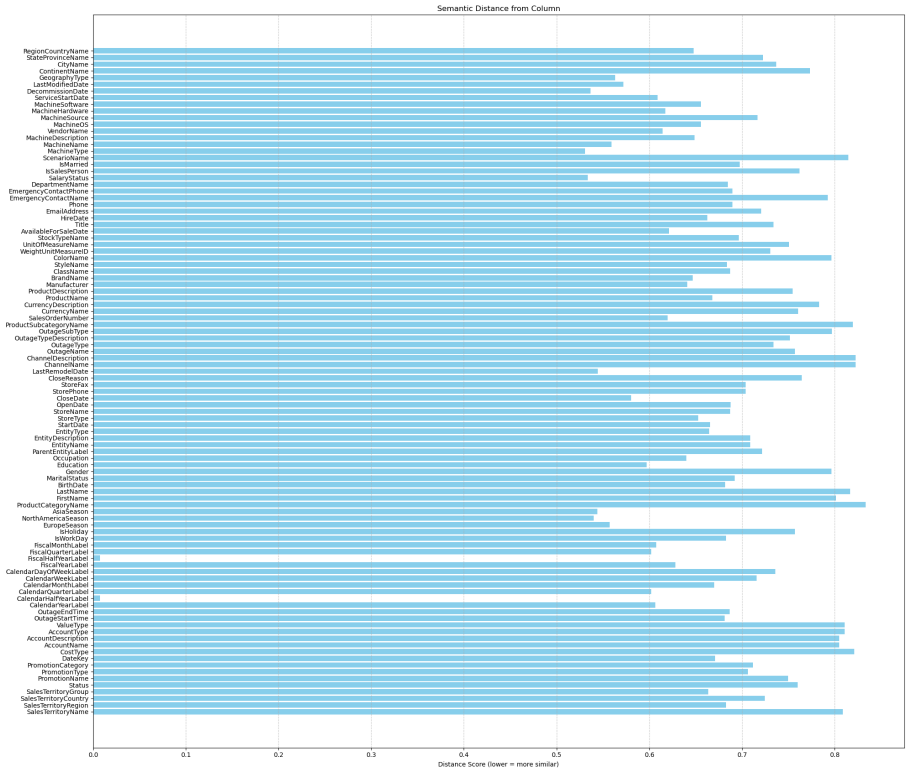

Figure B5: Cosine Semantic Distance from column "CalendarHalfYearLabel", Database C

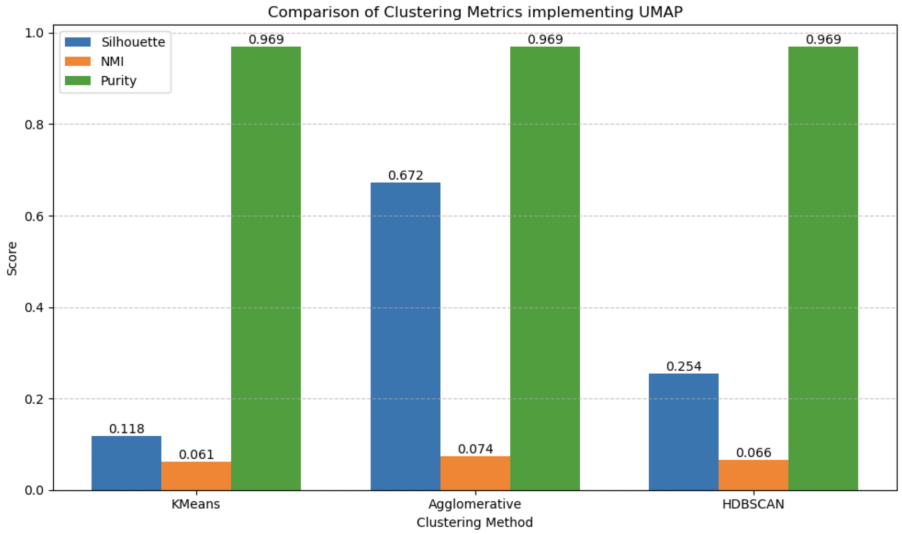

Figure B6: Comparison of Clustering Metrics Implementing UMAP before optimization, Database C

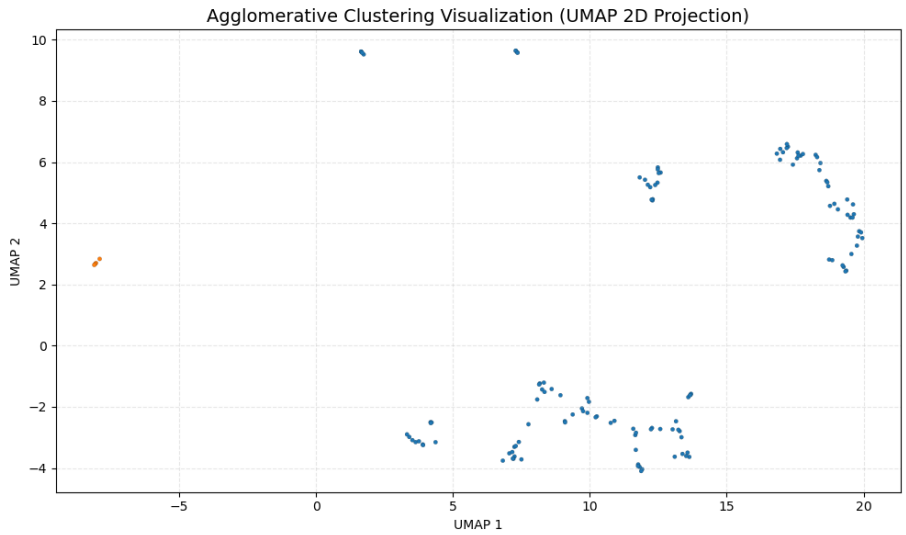

Figure B7: 2D representation of Clusters with original hyperparameters, Database C

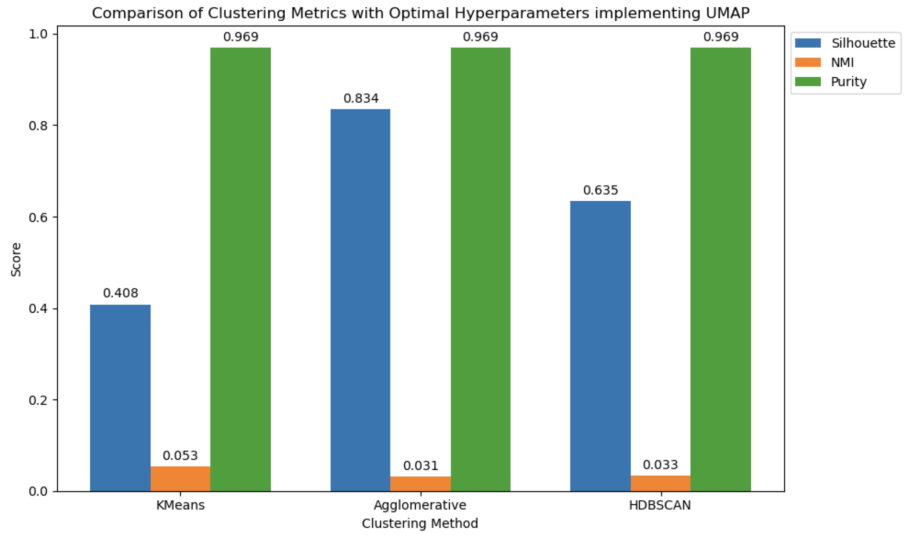

Figure B8: Comparison of Clustering Metrics Implementing UMAP after optimization, Database C

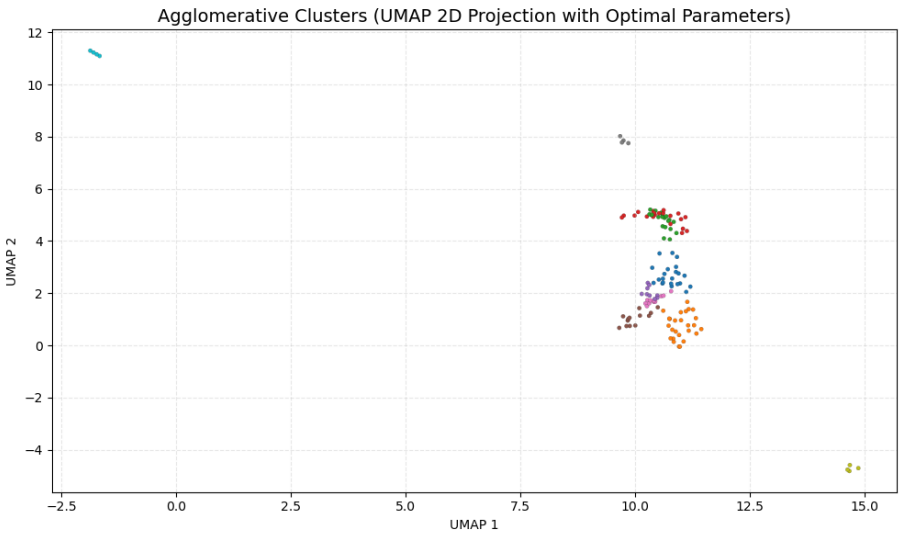

Figure B9: 2D representation of Clusters with optimized hyperparameters, Database C

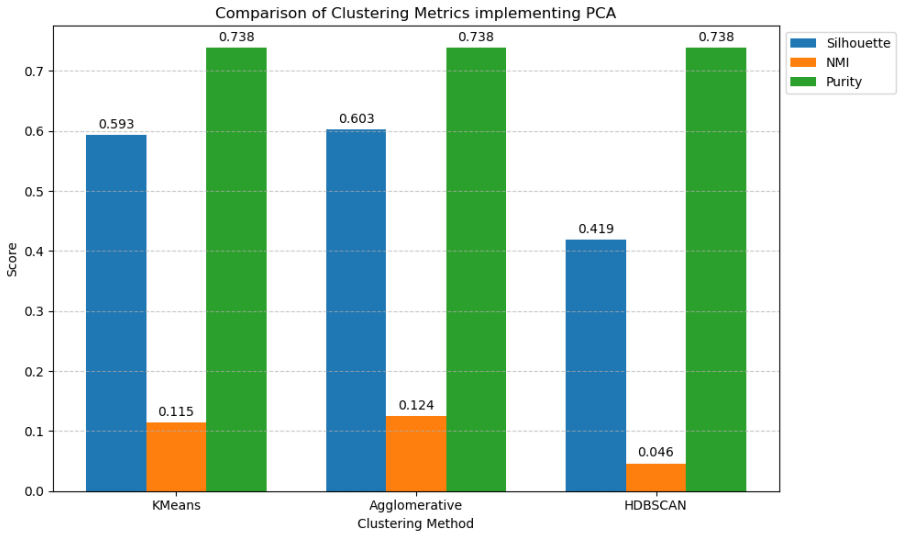

Figure B10: Comparison of Clustering Metric Implementing PCA before optimization, Database A

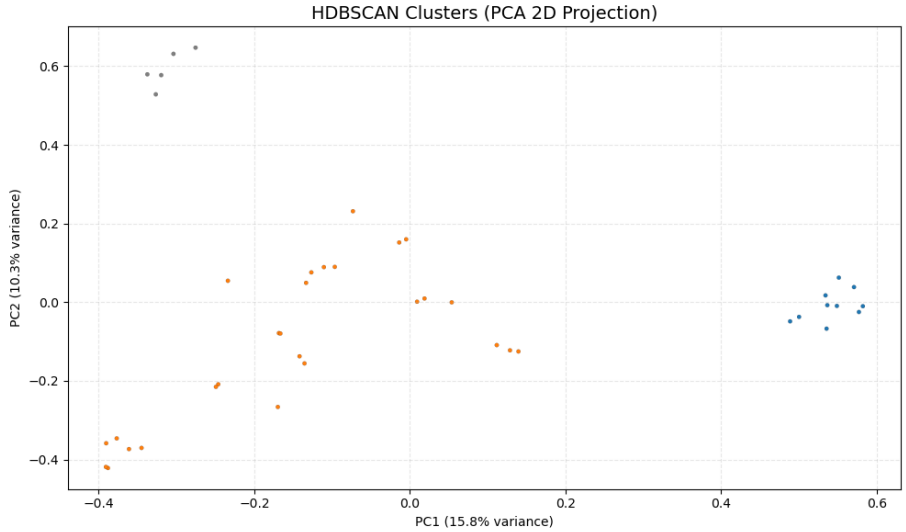

Figure B11: 2D representation of Clusters with original hyperparameters, Database A

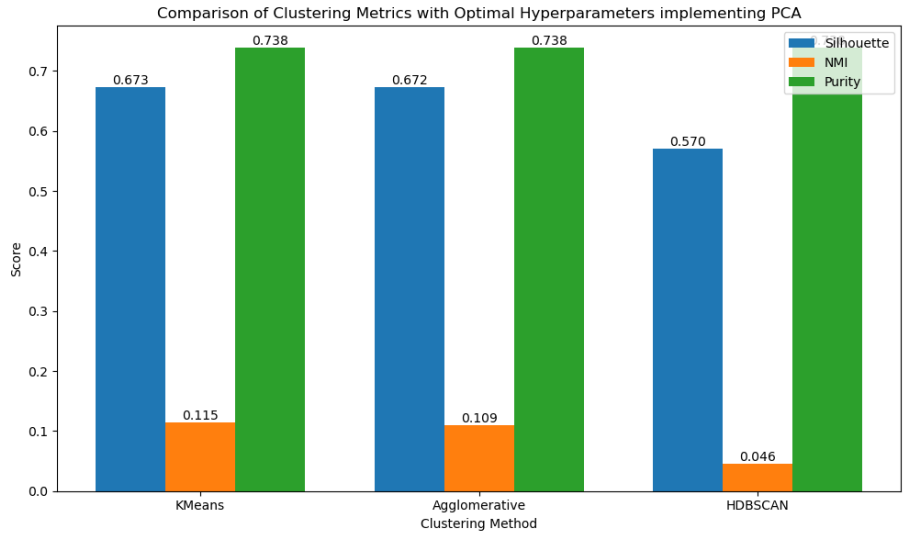

Figure B12: Comparison of Clustering Metric Implementing PCA after optimization, Database A

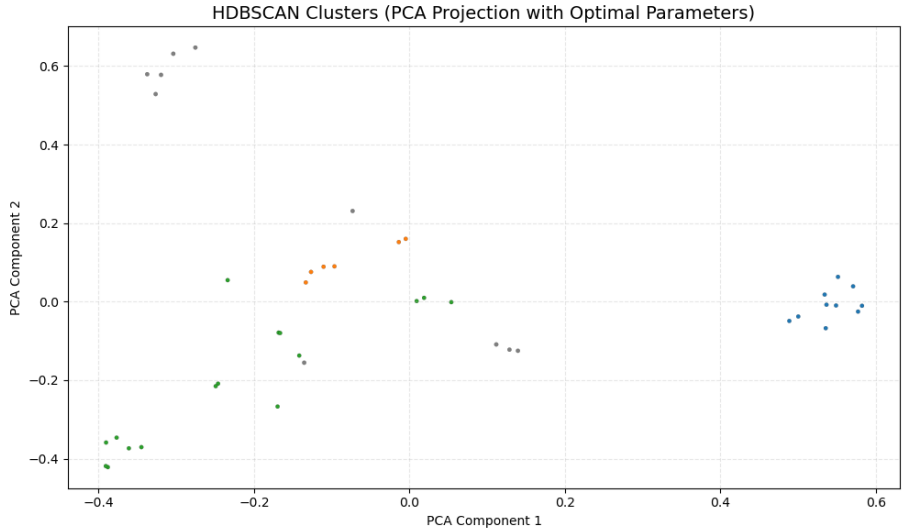

Figure B13: 2D representation of Clusters with optimized hyperparameters, Database A

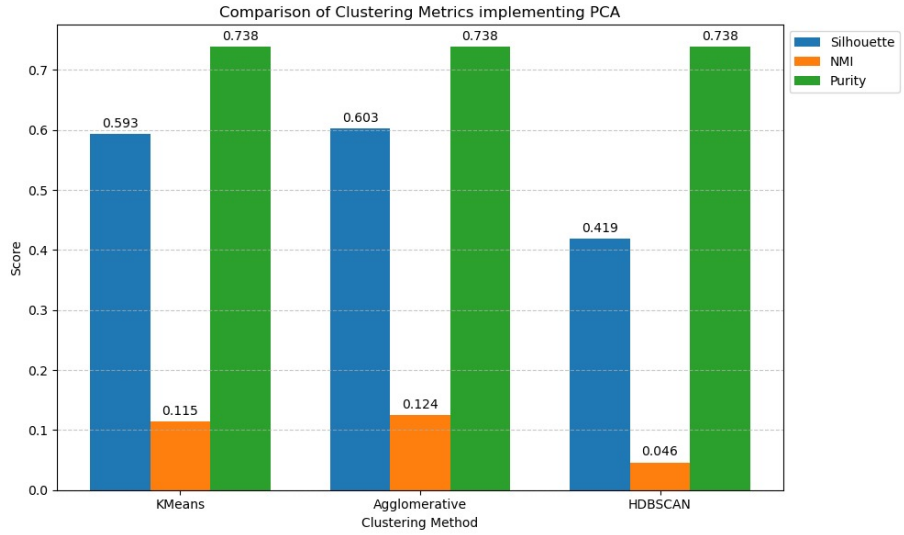

Figure B14: Comparison of Clustering Metric Implementing PCA before optimization, Database B

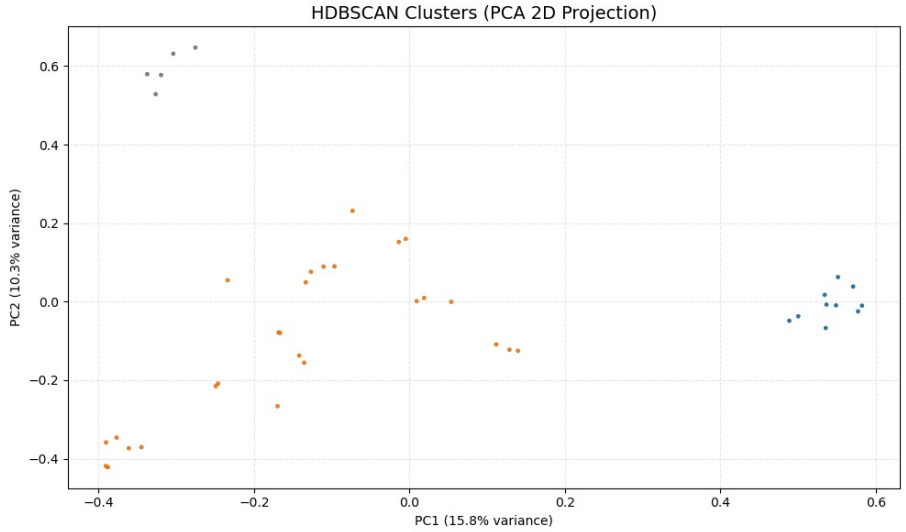

Figure B15: 2D representation of Clusters with original hyperparameters, Database B

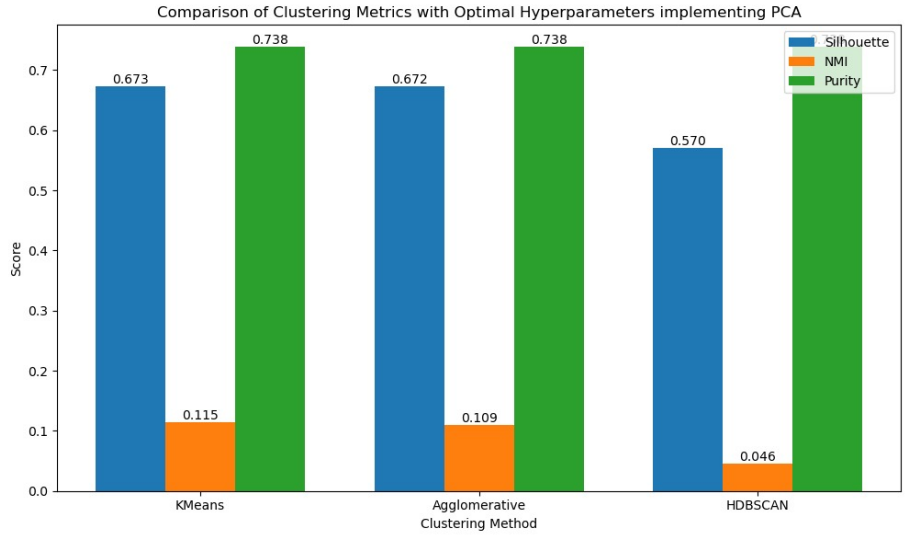

Figure B16: Comparison of Clustering Metric Implementing PCA after optimization, Database B

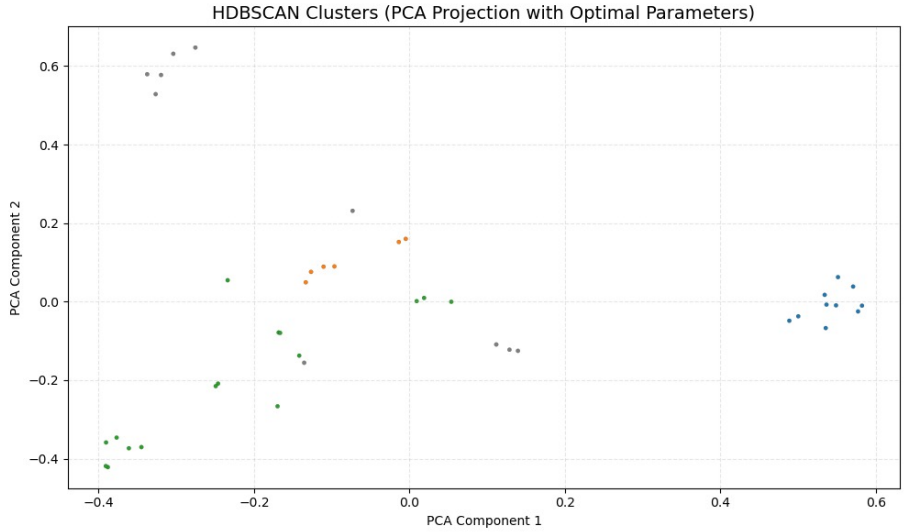

Figure B17: 2D representation of Clusters with optimized hyperparameters, Database B

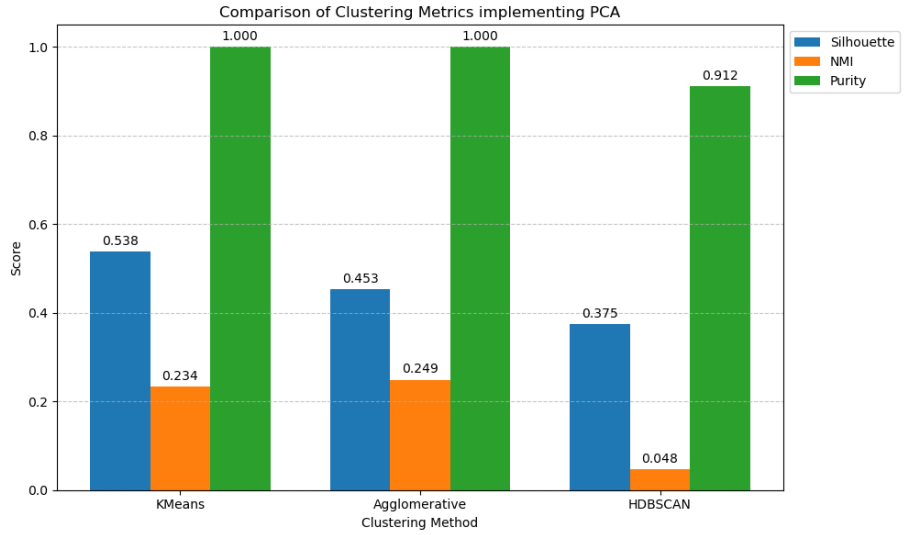

Figure B18: Comparison of Clustering Metric Implementing PCA before optimization, Database C

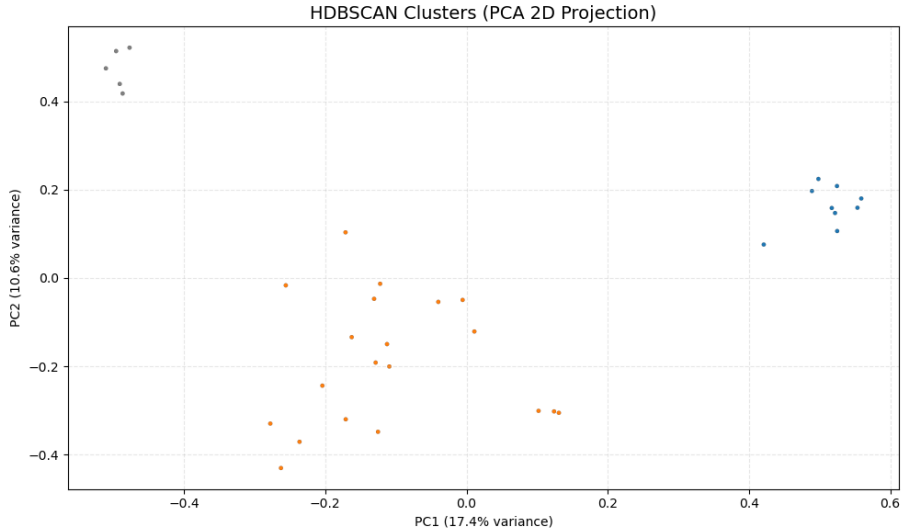

Figure B19: 2D representation of Clusters with original hyperparameters, Database C

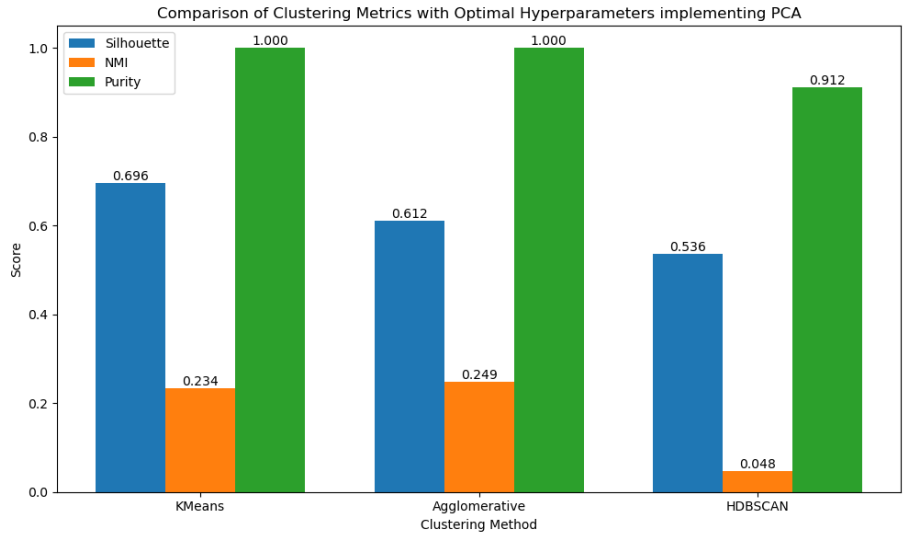

Figure B20: Comparison of Clustering Metric Implementing PCA after optimization, Database C

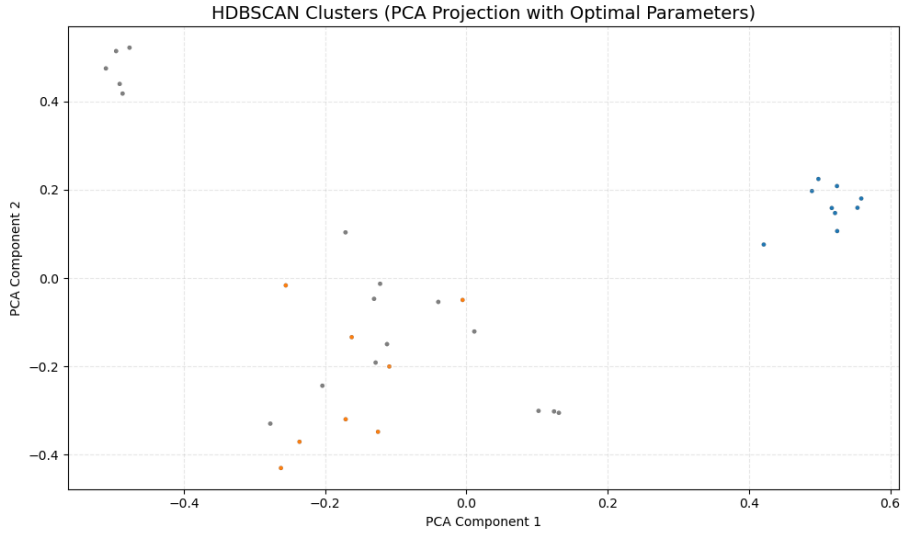

Figure B21: 2D representation of Clusters with optimized hyperparameters, Database C

Appendix C: AI Statement

The authors acknowledge the use of AI tools for the development of this thesis. `OpenAI-text-embeddings-3-small` model was employed to generate semantic embeddings for column classification, thanks to the contribution of Adage, which also allocated a budget for this purpose. Moreover, ChatGPT (OpenAI, GPT-4 architecture) was utilized during this thesis for the following purposes:

- Debugging and optimizing the existing code (e.g., memory efficiency).
- Hyperparameter optimization: assisted in designing grids to evaluate the clustering parameters, suggested color palettes for the visualization and interpretability of the clusters.
- Textual Refinement: improved grammatical clarity and check cohesion.

All outputs were reviewed, manually verified and have been adapted by the authors. ChatGPT served as an assistive tool; the final analysis and conclusions remain human-generated.