Revista Facultad de Ingeniería (Rev. Fac. Ing.) Vol. 34, No. 74.
L-ISSN: 0121-1129, e-ISSN: 2357-5328.

DOI: 10.19053/01211129.v34.n74.2025.20737

INTEGRATION OF THE K-MEANS ALGORITHM INTO POSTGRESQL THROUGH PL/PYTHON EXTENSIONS: A MODERATELY COUPLED ARCHITECTURE

Integración del algoritmo K-means en PostgreSQL mediante extensiones en PL/Python: una arquitectura medianamente acoplada

Fernando-Mauricio Vallejo-Cabrera
M.Sc. Universidad de Nariño, Pasto, Colombia.
fmvallejoc@udenar.edu.co

Ricardo Timarán-Pereira
Ph.D. Universidad de Nariño, Pasto, Colombia.
ritimar@udenar.edu.co

Anivar Chaves-Torres
Ph.D. Universidad de Nariño, Pasto, Colombia.
anivarchaves@udenar.edu.co

Received: 20-06-2025
Accepted: 16-12-2025
Available online: 31-12-2025

ABSTRACT

This paper presents the process of integrating the K-means partitioning clustering algorithm into the PostgreSQL database management system by developing an extension in PL/Python. The proposal is framed within a moderately coupled architecture, aiming to incorporate machine learning capabilities directly into the DBMS. This integration allows clustering processes to be executed without the need to rely on external tools, thus facilitating access to data mining techniques for small and medium-sized organizations and contributing to the generation of knowledge from large volumes of data, with a view to improving their competitiveness. The research is focused on extending PostgreSQL by incorporating the descriptive technique of clustering, specifically the K-means algorithm, to segment datasets into homogeneous groups. To achieve this, the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology was adopted, which is suitable for data mining projects and comprises six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. As a result, an extension was designed, developed, and implemented to expand PostgreSQL's functionalities that made it possible to execute clustering processes efficiently. The solution was validated through comparative tests against specialized tools such as Weka and KNIME, showing consistent and high-quality results in data grouping. The extension maintains lower response times compared to WEKA and KNIME, and its performance is optimal for high data volumes and clustering exercises of greater dimensional complexity. In conclusion, the developed extension demonstrates that it is possible to enhance PostgreSQL's analytical capabilities through the integration of machine learning algorithms. This proposal represents a viable and accessible alternative for organizations that need to perform advanced data analysis without depending on external platforms.

Keywords: clustering algorithms; data analysis; data mining; databases; machine learning; Python; unsupervised learning

How to cite: F. M. Vallejo-Cabrera; R. Timarán-Pereira & A. Chaves-Torres. "Integration of the K-means Algorithm into PostgreSQL through PL/Python Extensions: A Moderately Coupled Architecture". *Revista Facultad de Ingeniería*, vol. 34, no. 74, e20737, 2025. https://doi.org/10.19053/01211129.v34.n74.2025.20737

RESUMEN

Este artículo presenta el proceso de integración del algoritmo de *clustering* particional k-means en el sistema gestor de bases de datos PostgreSQL, mediante el desarrollo de una extensión en PL/Python. La propuesta se enmarca en una arquitectura medianamente acoplada, con el objetivo de incorporar capacidades de aprendizaje automático directamente en el SGBD. Esta integración permite ejecutar procesos de *clustering* sin necesidad de recurrir a herramientas externas, facilitando así el acceso a técnicas de minería de datos para pequeñas y medianas organizaciones, y contribuyendo a la generación de conocimiento a partir de grandes volúmenes de datos con miras a mejorar su competitividad. La investigación se orienta a extender PostgreSQL mediante la incorporación de la técnica descriptiva de *clustering*, específicamente el algoritmo k-means, para segmentar conjuntos de datos en grupos homogéneos. Para ello, se adoptó la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), adecuada para proyectos de minería de datos. Esta comprende seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación e implementación. Como resultado, se diseñó, desarrolló e implementó una extensión que amplía las funcionalidades de PostgreSQL, permitiéndole ejecutar procesos de *clustering* de manera eficiente. La solución fue validada mediante pruebas comparativas con herramientas especializadas como Weka y KNIME, evidenciando resultados consistentes y de alta calidad en la agrupación de datos. La extensión mantiene tiempos de respuesta inferiores frente a WEKA y KNIME y su rendimiento es óptimo para volúmenes de datos elevados y ejercicios de *clustering* de mayor complejidad dimensional. En conclusión, la extensión desarrollada demuestra que es posible potenciar las capacidades analíticas de PostgreSQL mediante la integración de algoritmos de aprendizaje automático. Esta propuesta representa una alternativa viable y accesible para organizaciones que requieren realizar análisis de datos avanzados sin depender de plataformas externas.

Palabras clave: algoritmos de clustering; análisis de datos; aprendizaje automático; aprendizaje no supervisado; bases de datos; minería de datos; Python.

INTEGRAÇÃO DO ALGORITMO K-MEANS NO POSTGRESQL POR MEIO DE EXTENSÕES EM PL/PYTHON: UMA ARQUITETURA MODERADAMENTE ACOPLADA

RESUMO

Este artículo presenta el proceso de integración del algoritmo de *clustering* particional k-means en el sistema gestor de bases de datos PostgreSQL, mediante el desarrollo de una extensión en PL/Python. La propuesta se enmarca en una arquitectura medianamente acoplada, con el objetivo de incorporar capacidades de aprendizaje automático directamente en el SGBD. Esta integración permite ejecutar procesos de *clustering* sin necesidad de recurrir a herramientas externas, facilitando así el acceso a técnicas de minería de datos para pequeñas y medianas organizaciones, y contribuyendo a la generación de conocimiento a partir de grandes volúmenes de datos con miras a mejorar su competitividad. La investigación se orienta a extender PostgreSQL mediante la incorporación de la técnica descriptiva de *clustering*, específicamente el algoritmo k-means, para segmentar conjuntos de datos en grupos homogéneos. Para ello, se adoptó la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), adecuada para proyectos de minería de datos. Esta comprende seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación e implementación. Como resultado, se diseñó, desarrolló e implementó una extensión que amplía las funcionalidades de PostgreSQL, permitiéndole ejecutar procesos de *clustering* de manera eficiente. La solución fue validada mediante pruebas comparativas con herramientas especializadas como Weka y KNIME, evidenciando resultados consistentes y de alta calidad en la agrupación de datos. La extensión mantiene tiempos de respuesta inferiores frente a WEKA y KNIME y su rendimiento es óptimo para volúmenes de datos elevados y ejercicios de *clustering* de mayor complejidad dimensional. En conclusión, la extensión desarrollada demuestra que es posible potenciar las capacidades analíticas de PostgreSQL mediante la integración de algoritmos de aprendizaje automático. Esta propuesta representa una alternativa viable y accesible para organizaciones que requieren realizar análisis de datos avanzados sin depender de plataformas externas.

Palavras-chave: algoritmos de clustering; análise de dados; aprendizado de máquina; aprendizado não supervisionado; bancos de dados; mineração de dados; Python.

1. INTRODUCTION

According to specialized literature, data mining or Knowledge Discovery in Databases (KDD) is a technical process applied to large volumes of data for identifying underlying patterns, rules, and trends through the implementation of specific algorithms [1,2]. This discipline is structured around two main approaches: the predictive approach, which uses supervised learning to estimate the value of a target variable in unseen instances; and the descriptive approach, which relies on unsupervised learning for discovering and defining intrinsic and interesting relationships within the dataset [3].

Data mining is a crucial stage within the KDD cycle, where the utility of the resulting models often requires a contextual and subjective assessment by the user. The underlying Machine Learning algorithms typically comprise three fundamental components [4]: 1) the model, which defines the parameters to be adjusted by the input data; 2) the preference criterion used for the selection and comparison of alternative models; and 3) the search algorithm, which describes the iterative procedure (steps) to find the best model parameters.

Within the unsupervised learning paradigm, where training data lack prior labels [5], algorithms seek to infer the under-lying structure of the data by identifying patterns. These techniques are primarily applied to clustering and dimensionality reduction [6]. Clustering is an unsupervised learning approach where predictor variables do not know target variables [7], it is therefore an essential technique in descriptive data mining that seeks to partition a dataset into groups (clusters), in which the members of each group share a similar pattern or characteristic. Clustering techniques are divided into hierarchical and partitional methods. While hierarchical clustering provides a more detailed structural coherence, partitioning clustering has become a more demanded option in Big Data environments because it requires fewer computational resources and allows efficient grouping even with large-sized datasets [8].

The k-means algorithm stands out among centroid-based partitioning clustering algorithms. It is widely recognized for its simplicity and computational efficiency, although it is sensitive to centroid initialization and it is primarily designed for numerical data [8]. Its purpose is to partition n observations into k clusters; therefore, each observation is classified into the cluster with the nearest mean [9]. K-means was chosen for this research because of its balance between performance and versatility in the exploratory analysis of massive data.

In the modern organizational context, the explosive growth in data volumes and databases has outpaced traditional analytical methods, thus making the analysis process a tedious and nearly impossible task [10]. Managing large volumes of information and discovering patterns have become essential for data-driven decision-making. While proprietary software solutions, such as those offered by Microsoft and Oracle, integrate data mining capabilities that enhance response times and security [1], they represent a high license cost that is often unaffordable for Micro, Small, and Medium Enterprises.

In contrast, the PostgreSQL Database Management System (DBMS) emerges as a robust alternative with a BSD license (open source). Its nature allows unrestricted use, modification, and distribution, thus aligning with the cost-saving needs of Micro, Small, and Medium Enterprises. However, PostgreSQL lacks a native implementation of advanced data mining algorithms. To overcome this limitation and capitalize on the advantages of its open license, it is necessary to extend its functionality.

This research proposes the development and implementation of an extension for PostgreSQL, using Python's procedural language - PL/Python. This approach, classified as a moderately coupled architecture [11], involves the compilation of functions that allow the database engine to execute

additional analytical processes. The choice of Python is strategic, given its simple syntax, its robustness in data science, and the availability of powerful libraries such as NumPy, Pandas, and, fundamentally, Scikit-learn, which automate the handling of large volumes of data.

The adoption of moderately coupled architecture offers significant advantages in terms of portability (the extension can be installed in future versions of PostgreSQL without modifying the DBMS core) and provides a better balance between response time, processing, and data security compared to loosely coupled architectures (which depend on external applications and network connections) [11].

The integration of Machine Learning capabilities into PostgreSQL has been a topic of intense research, with studies exploring different architectures:

- **Integration of classification and rules**: Robles and Sotolongo [1] integrated decision trees (ID3) and induction rules (PRISM, 1R) into PostgreSQL and achieved improvements in response times through table partitioning. Montero and his team [2] also used a moderately coupled architecture to integrate association rules (Apriori, FP-Growth, EquipAsso), comparing performance and rule generation. More recently, Timarán and Chavez [12] proposed the integration of supervised techniques such as C4.5 using PL/pgSQL, emphasizing the need to validate performance with large datasets.
- **Loosely coupled architectures and algebraic operators**: Calderón et al [13]. developed Tariykdd, a loosely coupled tool for association and classification algorithms, although they acknowledged the performance penalty caused by external connection. In contrast, Timarán and Millán [14] adopted a strongly/natively coupled approach, extending the SQL language with new algebraic primitives for classification, seeking to solve the problems of scalability and performance associated with loosely coupled architectures.

Although significant efforts have been made to integrate data mining into PostgreSQL, most have focused on predictive techniques (classification and association rules). This article addresses a gap in literature by focusing on the descriptive technique of clustering, specifically the k-means algorithm implemented through a moderately coupled architecture in PL/Python. The main objective is to present research project results focused on the analysis, design, development, and implementation of an extension to efficiently integrate the k-means algorithm into the PostgreSQL DBMS, leveraging the potential of Python's data science libraries.

2. MATERIALS AND METHODS

This study is classified as applied research because it seeks to generate a practical solution to a technological problem: the lack of native clustering capabilities in the PostgreSQL DBMS. It adopts a quantitative and descriptive approach, focusing on characterizing the unsupervised clustering technique and documenting the design, development, and performance of the resulting extension. The methodological design is non-experimental, since the study does not manipulate variables but rather observes, implements, and evaluates the behavior of the k-means algorithm integrated into a relational data-base environment.

For management and execution of data mining tasks, the CRISP-DM standard was selected. This methodology, recognized for its iterative and robust lifecycle, is ideal for projects that integrate knowledge discovery with software engineering. The CRISP-DM model is flexible and can be customized easily because it makes it possible to create a data mining model that adapts to concrete needs [15]. Activities were structured according to the six phases of CRISP-DM, adapted to the task of integrating

the partitional k-means algorithm into PostgreSQL through extensions in PL/Python. Table 1 presents the activities carried out at each phase:

Table 1. Activities conducted at each phase of CRISP-DM

Phase	Activities and tasks
Business understanding	A systematic literature review was conducted to characterize the descriptive clustering technique, focusing on partitional algorithms. Fundamental information was gathered regarding the procedure, advantages, and disadvantages of k-means. Simultaneously, the PostgreSQL extension architecture was examined, along with the required functionality of Python libraries such as NumPy, Pandas, and Scikit-learn as essential tools for implementation.
Data understanding	This phase focused on familiarization with and identification of the technical requirements of the k-means algorithm. The input data structure required for the algorithm's execution was analyzed, as well as the output data structure (clusters and centroids) that must be exposed to PostgreSQL SQL environment.
Data preparation	A detailed design of the SQL/Python functions encapsulating the k-means algorithm was carried out. This included defining the input parameters, managing data type transformations between PostgreSQL and Python (conversion of tables into NumPy/Pandas arrays), and designing the return data structure (composite type or table).
Modeling	The extension was developed and integrated. Necessary functions were coded using the PL/Python language to invoke the optimized implementation of the k-means algorithm. Finally, these functions were integrated into PostgreSQL's extension framework by making the algorithm directly accessible through SQL commands.
Evaluation	Several datasets with diverse characteristics were selected to perform functionality tests and performance evaluation (benchmark) of the extension. The evaluation focused on the accuracy of the generated clustering and, crucially, on efficiency and resource consumption compared to executing the algorithm outside the DBMS.
Deployment	The final phase consisted of exhaustive documentation of the research process and the development of an installation and usage manual for the extension. Additionally, a public repository of the extension was established to facilitate its distribution, download, and adoption by interested Micro, Small, and Medium Enterprises, thereby achieving the objective of technological transfer.

3. RESULTS

3.1 Extension Design and Components

The design of the extension is based on a set of procedural SQL functions implemented in PL/Python that orchestrate the complete clustering workflow within the DBMS. This workflow encompasses from data ingestion and preprocessing to the execution of the k-means algorithm and the visualization of results. The persistence of intermediate and final data is managed through a set of dynamic auxiliary tables created within a dedicated schema named clustering, thus ensuring an isolated working environment.

The following section describes the functional components and auxiliary data structures that enable in-database integration.

3.1.1 Data Loading and Preprocessing Functions

These functions are designed to prepare the dataset for analysis by loading it into a standardized format compatible with the extension (Table 2).

Table 2. Data loading and preprocessing functions

PL/Python function	Description	Data flow and auxiliary persistence
load_table_py	It loads the records from an existing PostgreSQL table into the extension's working environment.	It stores the data in the temporary table *cl_data*. This table has a variable structure *(m × n)*, where *m* is the number of records and *n* is the number of original attributes.
load_file_py	It enables data ingestion from an external CSV file, transferring the data into the working environment.	It stores the data in the temporary table *cl_data (m × n)*, where *m* is the number of records and *n* is the number of original attributes.
preprocessing_py	It performs data transformation for the execution of *k-means*. Internally, it applies *normalization* to *numerical attributes* and *binarization (one-hot encoding)* to *categorical attributes*.	It stores the result in the temporary table *cl_data_pre*. Its structure is *m × p*, where *p (p ≥ n)* represents the number of attributes after the *binarization* of *categorical variables*.

3.1.2 Modeling and Result Generation Functions

These functions execute the k-means algorithm and expose the main analytical outputs to the user in tabular format (Table 3).

Table 3. Modeling and result generation functions

PL/Python function	Description	Data flow and auxiliary persistence
kmeans_py	Central function that executes the *k-means* algorithm using the preprocessed data stored in *cl_data_pre*, with parameters *(K, number of iterations, seed, etc.)* defined by the user.	The trained model is serialized and externally persisted in the file *model.pickle* for subsequent reuse.
result_py	It generates and enables visualizing clustering results, assigning a cluster *label* to each record in the original *dataset*.	It stores the data with the cluster assignments in the table *cl_result*. Its structure is *m × (n + 1)*, where the additional column corresponds to the *cluster label*.
centroids_py	It displays the final centroids of the *k-means model*.	It stores the *centroids* in the *cl_centroids* table. Its structure is *k x (n+1)*, where *k* is the number of clusters and the additional column is the *cluster label* of each *centroid*.
summary_py	It provides a statistical summary of the clustering by counting the number of data points in each *cluster*.	It stores the summary in the table *cl_summary (k × 3)*. The columns include the cluster label, the total count of grouped records, and the percentage of records with respect to the number of records in the dataset.

3.1.3 Model Evaluation Functions

Critical auxiliary functions have been implemented for selecting the optimal number of clusters (K), which are essential for model validation (Table 4).

Table 4. Model evaluation functions

PL/Python function	Description	Data flow and auxiliary persistence
inertia_py	It retrieves and displays the total inertia *(Within-Cluster Sum of Squares – WCSS)* and the *number of iterations* used in the execution of *kmeans_py*.	It does not use an auxiliary table; it returns the result directly.
elbow_py	Repeatedly executes the *k-means* algorithm varying the value of *K* within a defined range. The resulting output serves as the basis for the *elbow method* (graphic method) and the empirical determination of the optimal *K*.	It stores the results in the table *cl_elbow (q × 2)*, where *q* is the maximum value of *K* evaluated. Columns contain the *K* value and its associated *total inertia*.
silhouette_py	It calculates the *Silhouette Coefficient* for various values of *K*, a robust metric for evaluating the cohesion and separation of clusters. This value is also used to determine the optimal *K*.	It stores the coefficients in the table *cl_silhouette (q × 2)*, with columns for the *K* value and the corresponding average silhouette coefficient.

3.2 Functionality Test

To validate the functionality and accuracy of the k-means functions implemented in the PostgreSQL extension, the well-known Iris dataset was used (150 records, available at https://archive.ics.uci.edu/dataset/53/iris). This dataset is ideal for clustering tasks because it includes 50 samples of each of the three flower species: Iris-setosa, Iris-versicolor, and Iris-virginica. The dataset consists of the following properties measured in centimeters: Sepal length, Sepal width, Petal length, Petal width and Variety (Flower species label).

For the clustering test, the variety column was omitted from the algorithm's input, as it represents the class label to be inferred.

Initially, the implementation of the extension was verified by comparing its results with those obtained from standard data mining software. After executing the k-means algorithm with both, the PostgreSQL extension produced the same clustering results as those generated by the Weka tool, thus confirming the correct algorithmic implementation.

When comparing the clusters generated by the extension against the actual classification ("variety") of the original dataset, the following confusion matrix (Table 5) was generated, clusters were reordered to maximize correspondence and accuracy.

Table 5. Clustering confusion matrix

Real variety (label)	Cluster 0	Cluster 1	Cluster 2	Total variety	Accuracy by Class
Iris-setosa	0	**50**	0	50	100%
Iris-versicolor	**10**	0	**40**	50	80%
Iris-virginica	**42**	0	**8**	50	84%
Total records	52	50	48	150	

The analysis of Table 5 reveals the high effectiveness of the clustering process:

- **Iris-setosa**: All 50 records were correctly grouped into Cluster 1, achieving 100% accuracy.
- **Iris-versicolor**: 40 out of 50 records were as-signed to Cluster 2, resulting in 80% accuracy. The remaining 10 records were incorrectly grouped into Cluster 0.
- **Iris-virginica**: 42 out of 50 records were as-signed to Cluster 0, achieving 84% accuracy. The remaining 8 records were incorrectly grouped into Cluster 2.

The overall effectiveness of the model generated by the extension (sum of correct assignments over the total number of records: (50 + 40 + 42) / 150) was 88%, validating the implementation as an accurate tool for clustering analysis within PostgreSQL.

3.3 Extension Functionality Validation Tests

To validate and verify the accuracy of the extension, the Wine Quality dataset was used (available at: https://www.kaggle.com/datasets/taweilo/wine-quality-dataset-balanced-classification). This dataset is highly relevant for clustering tasks, as it contains 21,000 records and 12 numerical variables describing various physicochemical characteristics of wine. The attributes included in the dataset are fixed_acidity, volatile_acidity, citric_acid, residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density, pH, sulphates, alcohol, quality.

To ensure the reliability of the implementation, clustering results obtained through the developed extension were compared and cross-validated with those generated by standard and widely recognized data mining software, such as Weka and KNIME.

The following section presents the performance evaluation of the extension in comparison with established data mining tools (Weka and KNIME), focusing on how processing time behaves when key parameters vary:

3.3.1 Test 1: Performance when Data Volume Vary (Vertical Dimensionality)

This test aimed to understand how the solution scales with increasing record volume. Ten synthetic datasets were generated, with a volume ranging between 1,000 to 1,000,000 records, based on the 11 physicochemical variables from the Wine Quality dataset. The variables measured were:

Dependent Variable: Model construction time (measured in seconds).

- Independent Variables:
  - ◊ Number of records: varied from 1,000 to 1,000,000.
  - ◊ Number of clusters (k): varied between 2 and 10.
  - ◊ Number of attributes: remained constant at 11 attributes.

Results are shown in Figure 1.

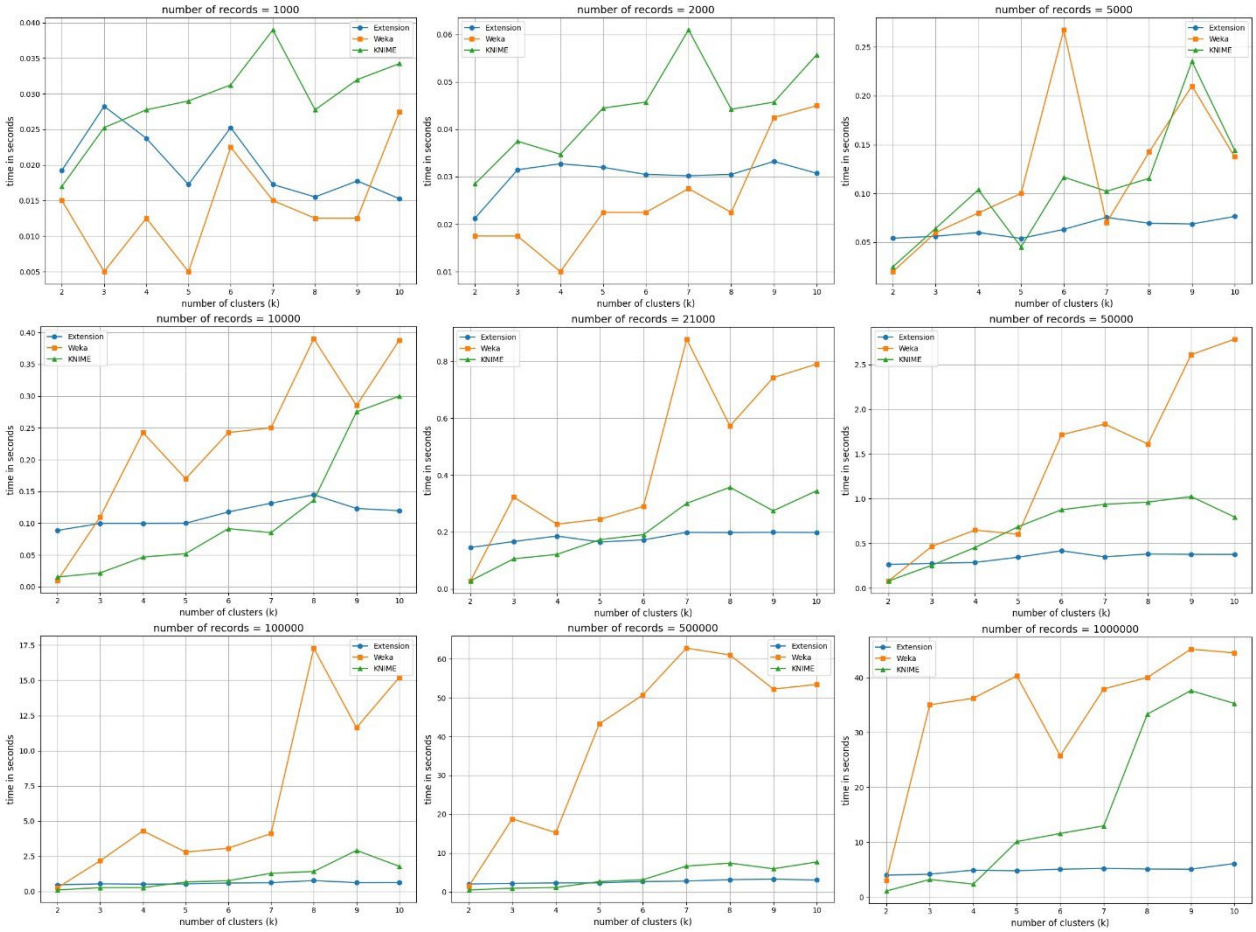

Figure 1. *Performance comparison varying the number of records and the value of k.*

3.3.2 Test 2: Performance when Horizontal Dimensionality Varies

This test aimed to understand the solution scales when increasing the number of variables (attributes). Ten datasets were created, each configured with a different number of attributes ranging from 2 to 11, based on the 11 physicochemical variables of the Wine Quality dataset.

This test is crucial for evaluating the efficiency of the extension when handling datasets with high feature dimensionality or many variables, a common scenario in data analysis.

This second performance test was designed to measure how the number of attributes (variables) affects processing time. To ensure a stable comparison baseline, the test used exclusively the Wine Quality dataset with 21,000 records, as in the previous test.

The goal was to determine the scalability of the K-means algorithm implemented in the extension in comparison with Weka and KNIME when the intrinsic complexity of the data increases.

Measured variables:

- **Dependent variable**: Model construction time (measured in seconds).
- **Independent variables**:
  - ◊ Number of records: Kept constant at 21,000 records.

◊ Number of clusters (k): Varied between 2 and 10 groups.

◊ Number of attributes: Gradually varied from 3 to 11 processed attributes.

Results are shown in Figure 2.

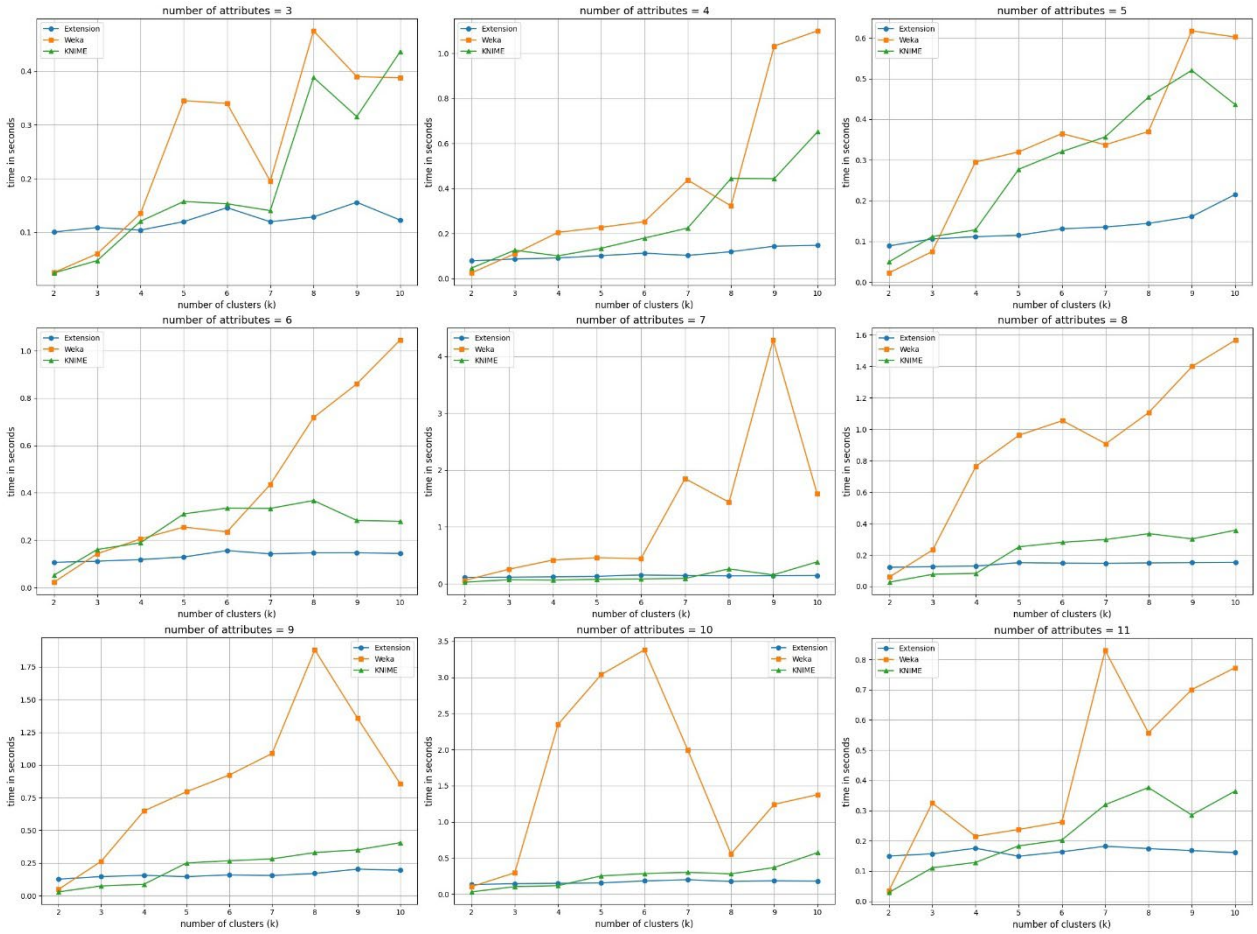

Figure 2. *Performance comparison varying the number of attributes and the value of k.*

4. DISCUSSION

When analyzing results (Figure 1), very clear performance patterns can be observed:

- **Weka**: It shows the best performance only on smallest datasets (1,000 and 2,000 records). However, as the data volume increases, its processing time rises sharply; therefore, it could be considered the worst-performing tool in scenarios involving high complexity or large datasets.

- **KNIME**: It provides better response times than Weka, especially in clustering tasks requiring fewer clusters (k = 2 to k = 4). However, its performance decreases noticeably as the value of the parameter k increases, thus indicating difficulty in scaling with a larger number of groups.

In conclusion from Test 1, the extension developed for PostgreSQL outperformed Weka and KNIME. Its advantage becomes especially significant in two scenarios:

1. When working with large datasets.

2. When the complexity of the clustering task increases; that is, when the number of groups to form (k) is higher.

In summary, while Weka excels only very low-complexity tasks, and KNIME performs effectively in low-to-medium complexity scenarios, the PostgreSQL extension is positioned as the most robust and efficient solution for complex and large-scale clustering tasks.

The analysis of Figure 2 reveals how an increase in horizontal dimensionality (number of attributes) impacts the performance of the evaluated tools, while keeping the data volume constant at 21,000 records.

Weka: It is confirmed as the worst performing tool in most scenarios. It only achieves the lowest response time when the clustering task presents the lowest combined complexity: a very small number of attributes and a low value of k (number of clusters).

KNIME: It generally offers intermediate performance. As in Test 1, it proves more efficient in low-complexity tasks with few clusters (low k). Its response time increases significantly when both the value of the parameter K and the number of attributes to be processed grow.

In conclusion, from Test 2, the developed PostgreSQL extension outperformed Weka and KNIME. Its advantage becomes evident and grows exponentially in more complex clustering tasks, that is, when working simultaneously with a larger number of attributes and a higher number of clusters to form. This positions the extension as the most scalable and efficient solution for data mining tasks involving high dimensionality.

5. CONCLUSIONS

This research achieved its objective by extending the capabilities of the PostgreSQL Database Management System, enabling the efficient execution of clustering processes directly within the engine. This goal was accomplished through the successful implementation of the k-means algorithm via an extension developed in PL/Python.

Functionality tests confirmed the correct operation of the extension and demonstrated that clustering results are highly comparable with the results obtained with specialized data mining tools such as Weka.

The well-known sensitivity of the k-means algorithm to centroid initialization is reaffirmed. It must be considered when interpreting the quality of the formed clusters. The comparative performance analysis against Weka and KNIME yielded conclusive results in terms of scalability (Table 6).

Table 6. Comparative performance analysis against Weka and KNIME.

Test metric	Performance conclusion
Vertical dimensionality (Number of records)	The extension maintains lower response times compared to the competitors. Its performance is optimal for large data volumes and large-scale clustering tasks.
Horizontal dimensionality (Number of variables)	For datasets with four or more variables, the developed extension proved to be more efficient, validating its usefulness in high-dimensional clustering scenarios.

This research represents a significant contribution to the PostgreSQL ecosystem, trans-forming it into a more robust database engine for advanced data analysis and data mining by facilitating the execution of descriptive algorithms in-database.

To consolidate and expand the impact of this research, the following work paths are proposed:

1. Optimization and validation

- Validation in real environments: Conduct exhaustive performance tests using real and production datasets to confirm the extension's efficiency and identify opportunities for optimization in real-world scenarios.
- Processing optimization: Explore the use of table partitioning strategies to handle large datasets more efficiently, improving response times and reducing resource consumption.
- Scalability and parallelization: Investigate and implement parallel processing techniques to optimize algorithm execution, ensuring adequate performance for even larger volumes of data.

2. Functional expansion

- Diversification of clustering algorithms: Incorporate other clustering approaches, such as hierarchical clustering and density-based clustering (DBSCAN) to provide more adaptable alternatives for diverse data structures and types.
- Integration of descriptive techniques: Extend PostgreSQL's capabilities by implementing methods such as association rules, enabling the discovery of useful patterns within the stored data.
- Development of predictive techniques: Integrate classification and regression algorithms to enhance PostgreSQL's capabilities in building predictive models and supporting data-driven decision making.

3. Usability

- Development of a graphical interface: Implement a visual platform to facilitate the use of the developed extensions, improving user interaction and providing a clearer and more intuitive representation of clustering and data mining results.

AUTHORS' CONTRIBUTION

Fernando-Mauricio Vallejo-Cabrera: Investigation, data curation, validation, writing – original draft.

Ricardo Timarán-Pereira: Conceptualization, supervision, methodology, riting – review & editing.

Anivar Chaves-Torres: Formal analysis, methodology, writing – review & editing.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

FUNDING

No external funding was received for this study.

REFERENCES

[1] Y. R. Aranda, A. R. Sotolongo, "Integración de los algoritmos de minería de datos 1R, PRISM e ID3 a PostgreSQL," *Journal of Information Systems and Technology Management*, vol. 10, no. 2, pp. 389-406, 2013. https://doi.org/10.4301/s1807-17752013000200012

[2] A. Montero, J. M. Reyes, J. C. Díaz, "Integrando minería de reglas de asociación en PostgreSQL," *Ingeniería Solidaria*, vol. 15, no. 11, pp. 24-36, 2022.

[3] M. A. García Vico, P. González García, C. J. Carmona, "Modelos descriptivos basados en aprendizaje supervisado para el tratamiento de grandes volúmenes de datos y flujos continuos de datos," en *Actas de la XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA)*, Granada, España, 2018, pp. 1402–1407. https://sci2s.ugr.es/caepia18/proceedings/docs/CAEPIA2018_paper_293.pdf

[4] V. Valcárcel Asencios, "Data Mining y el descubrimiento del conocimiento," *Industrial Data*, vol. 7, no. 2, pp. 83-86, 2004. https://doi.org/10.15381/idata.v7i2.6140

[5] R. Saravanan, P. Sujatha, "A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification," in *Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, pp. 945-949, 2018. https://doi.org/10.1109/ICCONS.2018.8663155

[6] B. Mahesh, "Machine Learning Algorithms - A Review," *International Journal of Science and Research*, vol. 9, no. 1, pp. 381-386, Jan. 2020. https://doi.org/10.21275/ART20203995

[7] N. M. Mahfuz, M. Yusoff, Z. Idrus, "Clustering heterogeneous categorical data using enhanced mini batch K-means with entropy distance measure," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 1048-1059, Feb. 2023. https://doi.org/10.11591/ijece.v13i1.pp1048-1059

[8] S. Singh, S. Srivastava, "Review of Clustering Techniques in Control System," *Procedia Computer Science*, vol. 173, pp. 272-280, 2020. https://doi.org/10.1016/j.procs.2020.06.032

[9] J. Alzubi, A. Nayyar, A. Kumar, "Machine Learning from Theory to Algorithms: An Overview," *Journal of Physics: Conference Series*, vol. 1142, no. 1, e012012, 2018. https://doi.org/10.1088/1742-6596/1142/1/012012

[10] Y. Rivera, *Desarrollo de árboles de decisión como extensión al gestor de bases de datos PostgreSQL*, Grade Thesis, Universidad de Ciencias Informática, La Habana, Cuba, 2014. https://repositorio.uci.cu/jspui/handle/ident/9264

[11] R. Timarán, "Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de Bases de Datos: Un Estado del Arte," *Ingeniería y Competitividad*, vol. 3, no. 2, pp. 45-55, 2011. https://doi.org/10.25100/iyc.v3i2.2327

[12] R. Timarán, A. Chavez, "Integration of Supervised Machine Learning Techniques with Postgresql Dbms in a Middly Coupled Architecture," en *Encuentro Internacional de Educación en Ingenieria (EIEI ACOFI)*, 2023. https://doi.org/10.26507/paper.2982

[13] Á. F. Calderón-Romero, A. O. Y Ramírez-Freyre, I. D. Alvarado, J. C. Guevara-Unigarro, *Tariykdd: una herramienta genérica de descubrimiento de conocimiento en bases de datos débilmente acoplada con el SGBD PostgreSQL*, Grade Thesis, Universida de Nariño, Pasto, Colombia, 2007. http://sired.udenar.edu.co/id/eprint/5744

[14] R. Timarán, M. Millán, "New algebraic operators and SQL primitives for mining classification rules," *International Journal of Robotics Research*, vol. 30, no. 2, pp. 115–123, May 2015.

[15] C. Schröer, F. Kruse, J. M. Gómez, "A systematic literature review on applying CRISP-DM process model," *Procedia in Computer Science*, vol. 181, pp. 526-534, 2021. https://doi.org/10.1016/j.procs.2021.01.199