International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-20

https://doi.org/10.70917/ijcisim-2026-0977
Article

Hybrid K-Nearest Neighbors with Ant Colony Optimization for Securing data warehouses against inferences

Fatima Zohra Benazza [1], Djamila Hamdadou [2] and Ilyes Khennak [3,*]

[1] 1 Oran 1 Ahmed Ben Bella University, Oran, Algeria; benazza.fatima@edu.univ-oran1.dz
[2] Oran 1 Ahmed Ben Bella University, Oran, Algeria; hamdadou.djamila@edu.univ-oran1.dz
[3] Laboratory for Research in Artificial Intelligence, USTHB, Algiers, Algeria
* Correspondence author: ilyes.khennak@usthb.edu.dz

Abstract: Data Warehouses (DWs) are among the most powerful technologies for storing and managing large volumes of corporate data, which often include sensitive or confidential information. However, they remain vulnerable to inference attacks and insufficient access control mechanisms. In recent years, Artificial Intelligence (AI) techniques have become key tools for enhancing DW security, particularly for detecting and preventing unauthorized inferences. In this work, we propose a hybrid approach that combines the K-Nearest Neighbors (KNN) classification algorithm with the Ant Colony Optimization (ACO) metaheuristic to strengthen data warehouse security. The objective is to minimize inference risks by optimizing variable selection and improving the accuracy of sensitive data classification. The proposed ACO–KNN model was evaluated using a dataset of 1,000 SQL analytical queries generated by the IBM Db2 Query Manager, representing realistic decision-support workloads. Experimental results show that the hybrid model significantly outperforms traditional KNN and other metaheuristic-based methods in terms of prediction accuracy, convergence speed, and inference prevention capability. This demonstrates the model's potential for practical integration into Business Intelligence (BI) and OLAP environments, contributing to more secure and reliable analytical decision-making.

Keywords: Data warehouse; Data security; Data inference; Machine Learning; K-Nearest Neighbors; Meta-heuristic; Ant Colony Optimization.

1. Introduction

Business Intelligence (BI) is a continuous process that utilizes various technologies to collect, process, analyze, and present data from diverse sources, providing organizations with actionable insights to support informed decision-making. This empowers managers and teams across departments with a comprehensive understanding of company performance, enabling them to respond swiftly to market opportunities and risks, ultimately gaining a competitive advantage.

Business Intelligence (BI) is fundamentally supported by a specialized information system known as a Business Intelligence System (BIS), which differs from traditional transactional information systems. These BI systems comprise multiple components that were historically consolidated within a data warehouse. A data warehouse is a centralized repository of integrated and historically recorded data designed to support strategic decision-making through analytical processing. While existing data warehouse development tools primarily focus on storage structure, data warehouses are characterized by their integration of heterogeneous data sources. This integrated data, often proprietary and sensitive,

is used by executives to inform strategic decisions and thus requires robust access control mechanisms for security purposes [1].

Nowadays, the use of computer networks has become essential, and our world has migrated to interconnected networks through the Internet [2][3], which makes them a target of attacks that are multiplying day by day. Hence, it is necessary to analyze risks and implement measures to minimize the vulnerability of systems while auditing the information system. In other words, it is necessary to adopt a security policy whose goal is to ensure: (i) integrity, (ii) confidentiality, (iii) availability, and (iv) non-repudiation.

The study of related work on data warehouse security has allowed us to identify two classes of approaches: (i) approaches focusing on the security of operations, which address who has access and what operations are allowed, and (ii) approaches addressing inference prevention, which focus on preventing users from inferring protected data from accessible information.

Artificial Intelligence (AI) offers powerful solutions for such challenges by efficiently analyzing large amounts of data, recognizing patterns, and supporting decision-making. In particular, machine learning techniques can be used to predict and prevent inference attacks. Among them, the K-Nearest Neighbors (KNN) algorithm is a well-known supervised learning method that classifies new data points based on similarity with labeled training examples. Despite its simplicity, its performance depends on parameters such as the choice of k and the similarity metric.

Metaheuristics, on the other hand, are optimization techniques inspired by natural processes and are widely applied to complex problems. The Ant Colony Optimization (ACO) algorithm, in particular, is inspired by the foraging behavior of ants and has shown strong performance in solving combinatorial optimization problems.

Accordingly, in this paper, we propose to combine KNN and ACO to secure the data warehouse against inferences in a way that complements On-Line Analytical Processing (OLAP) functionalities. The proposed approach prevents precise inferences generated through aggregation queries (e.g., MAX, MIN, SUM). More precisely, KNN is used to classify query types and detect risks of inference, while ACO optimizes query handling for better protection. Compared to existing approaches based on data perturbation [4,5,6], our method relies on query history, thereby preserving data integrity and avoiding the heavy perturbation processing required after each warehouse update.

In this study, we evaluated the effectiveness of the proposed hybrid method using a dataset of queries that was manually reorganized to suit the requirements of each algorithm. The performance was assessed using accuracy, recall, precision, F-measure, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

The remainder of this paper is organized as follows. Section 2 surveys and discusses related work on data warehouse security, KNN, and ACO. Section 3 presents the proposed approach. Section 4 reports the experimental results. Finally, Section 5 provides conclusions and outlines areas for future research.

2. Related Work

Data warehouse security is a multifaceted challenge that demands a robust, multi-layered approach and constant vigilance. Organizations must prioritize data protection through a combination of best practices and appropriate security technologies to mitigate a range of threats, including unauthorized access, data modification, disclosure, and denial-of-service attacks. This section presents a review of existing research on data warehouse security, exploring key themes and highlighting emerging trends in the field.

A multi-phase methodology for designing data warehouse security was proposed in [7]. The phases of the methodology include preliminary analysis, design, logical modeling, physical modeling, and implementation. In the preliminary analysis phase, the authors defined two categories of security requirements: basic and advanced. Basic requirements consist of hiding cubes, cube faces, data details, or dimensions. Advanced features employ data-driven measures, including face concealment and dynamic rule application, to implement sophisticated protection strategies. Although this model comprehensively covers all data in a warehouse, it does not propose a systematic method for identifying security requirements.

Soler et al. [8] proposed a *Model Driven Approach (MDA)* for developing secure data warehouses, using the *Query View Transformation (QVT)* language to automate the transition from conceptual to logical levels. However, the MDA approach lacks a verification mechanism to ensure that conceptual constraints are correctly translated and consistently enforced.

A comparative study presented in [9] analyzed security features of commercial OLAP tools, concluding that physical security (of infrastructures and servers) alone is insufficient to ensure overall

warehouse protection. The authors emphasized logical and application-level security but did not address inference attacks, which can arise from legitimate query combinations.

Sung et al. [4] defined *data security* as maintaining privacy in cube cells while providing accurate and accessible query results. They proposed the *zero-sum perturbation method*, which adds random values to cube cells so that row and column sums remain zero. This technique preserves query accuracy but only handles *SUM* queries and introduces significant computation overhead.

Cuzzocrea et al. [5] proposed a framework for generating a secure cube A' from an original cube A, based on recognized security metrics[6]. Their framework selects specific dimensions and data regions with skewed distributions to satisfy security constraints efficiently. While it offers reduced computational cost, it does not effectively handle *SUM* and *AVG* queries—both common in OLAP systems.

Dwork [6] introduced the theoretical foundation of *differential privacy*, providing rigorous guarantees for protecting individual data contributions in aggregate analyses. While highly secure, differential privacy introduces noise that can reduce analytical accuracy and is challenging to apply in real-time OLAP environments.

Elkhadir et al. [10] enhanced *Linear Discriminant Analysis (LDA)* for intrusion detection by replacing sample means with geometric mean vectors, improving robustness on *KDDCup99* and *NSL-KDD* datasets. Their approach demonstrated higher detection accuracy than traditional LDA variants, highlighting the potential of geometric feature modeling in security analytics.

Almansob and Lomte [11] proposed a scalable *Intrusion Detection System (IDS)* using *Principal Component Analysis (PCA)* for dimensionality reduction and *K-Nearest Neighbors (KNN)* for classification. Their work effectively addressed high-dimensional data challenges but still required tuning to balance detection accuracy and runtime efficiency.

Benaddi et al. [12] introduced a *PCA-FC-KNN* hybrid model to detect intrusions by reducing irrelevant features and optimizing classification accuracy. Their experiments on large datasets showed reduced false alarms but also highlighted a decline in performance with increasing data volume, underlining the need for further optimization.

The comparative analysis in [13] studied three reduction techniques—PCA, ANN, and NLPCA—and evaluated four classifiers: *Decision Tree (DT)*, *Support Vector Machine (SVM)*, *K-Nearest Neighbor (KNN)*, and *Naïve Bayes (NB)*. They also introduced new evaluation metrics (CDM, SPDM, SNDM, F1DM) to assess differences between original and reduced datasets using *CIDDS-2017* and *NSL-KDD*. Their findings showed that while KNN performs well, its accuracy is sensitive to feature space quality and parameter selection—an issue addressed by optimization in hybrid models.

Xin and Wang [14] explored deep learning-based feature selection for intrusion detection, analyzing its advantages and disadvantages compared to traditional approaches. They proposed a novel selection method validated through extensive comparative experiments, achieving high detection accuracy but at the cost of increased computational complexity.

In engineering contexts, hybridization between machine learning and metaheuristics has also shown great promise. Shariati et al. [15] combined *ANFIS* with *PSO* and *GA* to predict angular shear connector behavior, achieving higher precision and faster convergence. Similarly, Mu'azu [16] applied metaheuristic-trained *ANN models* (CFOA, ESDA, HGSOA, SCA) to predict foundation settlement with minimal error. These studies highlight the general strength of hybrid intelligent systems in achieving efficient and accurate predictions.

In the cybersecurity domain, Maazalahi and Hosseini [17] improved K-means-based intrusion detection through metaheuristic optimization, enhancing accuracy but increasing computational cost for large datasets. Hu et al. [18] conducted a deep analysis of nature-inspired algorithms applied to intrusion detection in cloud, edge, and IoT environments, identifying challenges such as convergence stability and adaptive scalability.

Finally, Tran et al. [19] introduced *Data Guard*, a fine-grained purpose-based access control system for large-scale data warehouses. It supports row, column, and sub-cell masking with high precision and low runtime cost. However, its focus on masking and access control does not address dynamic inference detection.

Taken together, prior studies reveal key strengths and limitations. Many focus narrowly on specific aggregate functions (mainly SUM), lack runtime adaptability, or apply generic intrusion detection models not tailored to OLAP inference control. Others introduce strong privacy guarantees but sacrifice analytical accuracy or computational efficiency.

Our proposed *Hybrid K-Nearest Neighbors with Ant Colony Optimization (KNN–ACO)* framework addresses these gaps. KNN classifies query patterns to detect potential inference risks, while ACO dynamically optimizes query handling to minimize security exposure. Unlike perturbation or masking methods, our approach preserves original data integrity and supports multiple aggregate types (SUM,

MAX, MIN). It bridges preventive and adaptive strategies, achieving a balance between security, accuracy, and computational efficiency, positioning it as a comprehensive and intelligent defense mechanism for data warehouse inference protection.

3. Background

In this section, we will discuss the foundational concepts underpinning the application of the K-Nearest Neighbors (KNN) algorithm in conjunction with Ant Colony Optimization (ACO) to enhance the security of data warehouses. We will first provide a comprehensive overview of the KNN algorithm, highlighting its effectiveness and prevalence in classification and anomaly detection tasks. Following this, we will delve into the principles of ACO, which is inspired by the foraging behavior of ant colonies, and examine its utility in optimizing processes. This integrated approach not only capitalizes on the strengths of both methodologies but also addresses critical challenges in data security, offering a robust framework for the protection of data assets.

3.1. Data Warehouse security

In this section, we present our approach for preventing inference attacks. These algorithms utilize the following data structures.

1) K-Nearest Neighbors

- **Machine learning** is a discipline of artificial intelligence that enables computers to learn from data without explicit programming. It is based on the ability of systems to learn models from data and make decisions based on these models. Supervised machine learning involves training an algorithm on a set of labeled data, where the responses are known, to predict outputs based on new inputs. Unsupervised learning, on the other hand, deals with unlabeled data to discover hidden structures or patterns. Classification is a supervised learning task aimed at predicting the class to which a new data item belongs on the basis of labelled examples, while regression predicts continuous values as a function of input variables. K-Nearest Neighbors (KNN) is a supervised classification algorithm. It works by comparing the similarity between new data and training data, selecting the K nearest neighbors, and predicting the majority class among these neighbors for the new data. KNN is simple but may require a good choice of parameters to achieve good performance, particularly on moderately sized datasets with a clear clustering structure. Algorithm 1 presents the detailed

3.1.1. Algorithm 1: K-Nearest Neighbors (KNN)

1. Input

- Training dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where x_i are feature vectors and y_i are class labels.
- A query instance q to be classified.
- Number of neighbors K to consider.

2. Distance Calculation

- For each $x_i \in D$, compute the distance $d(x_i, q)$.
- Common choice: Euclidean distance [20]

$$d(x_i, q) = \sqrt{\sum_j (x_{ij} - q_j)^2}.$$

- Other distance measures can be applied depending on the problem.

3. Selection of K Nearest Neighbors

- Sort training instances in ascending order of $d(x_i, q)$.
- Select the K nearest neighbors, denoted as $N_K(q)$.

4. Majority Vote

- Determine the most frequent class among $\{y_i : x_i \in N_K(q)\}$.
- Assign this class to q.

5. Output

- Predicted class label $\hat{y}$ for the query instance q.

The KNN algorithm is relatively simple to implement and understand, but it is important to choose the value of K wisely to achieve good performance. It is also essential to normalize the data if the features have different scales to avoid any bias in the calculation of distances.

2) Ant Colony Optimization

- **Discrete and continuous problems** Discrete problems are problems where the possible solutions are distinct and separate elements. For example, in a timetabling problem, schedules can be divided into distinct intervals and cannot be split. Discrete problems require specific approaches to be solved. Continuous problems, on the other hand, involve solutions that can be any real number in a continuous interval. For example, in the optimization of mathematical functions, the variables can take continuous values. Continuous problems require continuous optimization techniques to be solved. Metaheuristics are generic methods for solving difficult optimization problems where classical approaches are not effective. They are designed to explore the search space intelligently, often by combining heuristics to find high-quality solutions in a reasonable time. Metaheuristics include techniques such as Evolutionary Algorithms, Ant Colony algorithms, Simulated Annealing, Tabu Search and so on. They are often used to solve both discrete and continuous problems. There are a limited number of metaheuristics commonly used to solve discrete problems, including Genetic Algorithms, Ant Colony Algorithms, Tabu Search, etc. These methods have been adapted to deal with discrete problems by exploring solutions in a discrete search space.
- **Ant Colony Optimization** Ant Colony Optimization (ACO) is a metaheuristic technique inspired by the behavior of ants for solving optimization problems. By employing virtual agents known as artificial ants that adhere to basic rules, this approach seeks to discover effective solutions for a range of optimization problems. ACO algorithms are classified within a distinct group of metaheuristics that leverage cooperation and communication principles to direct the quest for optimal solutions [21]. As mentioned earlier, the ACO algorithm was designed after analyzing the behavior of social insects, particularly ants. Ants behave collectively, putting the well-being of the community first. Each ant acts autonomously, without direct supervision [22]. Ants communicate using volatile chemical substances known as pheromones to mark their paths for others to follow. A typical example is the trail from the nest to a food source: when an ant finds food, it releases pheromones on its way back to the nest using glands in its abdomen. This scented trail guides other ants to the food source and back to the nest. This collective behavior, where individuals indirectly support each other, enhances the overall efficiency of the community. This observation, seen in ant colonies, inspired the Deuneubourg bridge experiment, which led to the development of the Ant Colony Optimization (ACO) algorithm [22]. Multiple variations of the ACO algorithm have since been developed [23, 24], demonstrating its versatility. The first ACO was developed by Dorigo et al. [25] and named Ant System (AS). An artificial ant differs considerably from a natural or biological ant. An artificial ant has a memory implemented as a list of previously visited vertices. This guarantees that no ant will revisit the same vertex, and this memory is erased once the ant has completed its cycle. Algorithm 2 presents the detailed Ant Colony Optimization:

3.1.2. Algorithm 2: Ant Colony Optimization (ACO)

1. Input

- A graph G = (V, E) representing the problem.
- Number of ants m.
- Parameters: pheromone importance α, heuristic importance β, evaporation rate ρ, and number of iterations T .

2. Initialization

- Initialize pheromone levels τ_{ij} on all edges $(i, j) \in E$.

3. Construct Solutions

- For each ant, build a solution by moving from node to node.
- The probability of choosing edge (i, j) is given by:

$$P_{ij}^{k} = \frac{(\tau_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{(i,l) \in N_i} (\tau_{il})^{\alpha} \cdot (\eta_{il})^{\beta}} \quad (1)$$

where η_{ij} is the heuristic information and N_i is the neighborhood of node i.

4. Update Pheromones

- Evaporate pheromones on all edges:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij}$$

- Deposit pheromones based on solution quality:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta \tau_{ij}^{k}$$

where $\Delta\tau_{ij}^{k}$ is proportional to the quality of the solution built by ant k.

5. Stopping Criterion

- Repeat steps 3 and 4 until the maximum number of iterations T is reached or convergence is achieved.

6. Output

- Return the best solution found among al.

4. KNN and ACO for securing data warehouse against inferences

The proposed approach aims to enhance the security of the data warehouse by preventing unauthorized inferences through a robust pipeline that combines metaheuristic optimization and machine learning techniques. Initially, the data is collected and extracted using advanced query mechanisms. The dataset is then manually adapted to ensure compatibility with metaheuristic algorithms. Using ant colony optimization (ACO), the most critical variables are identified, reducing the dataset to its essential features while preserving its integrity. This refined data set is used to train a K-Nearest Neighbors (KNN) model, which classifies incoming user requests. The system predicts whether a request constitutes an inference, which could expose sensitive information, or a valid query. Requests identified as inferences are rejected to protect the data warehouse from potential security breaches. By leveraging the strengths of metaheuristic algorithms for feature selection and the predictive power of the KNN model, the proposed approach provides a proactive defense mechanism, ensuring the secure handling of user requests and safeguarding the integrity of the data warehouse.

- **Solution encoding** To adapt Ant Colony Optimization (ACO) for our problem, we drew inspiration from the Traveling Salesperson Problem (TSP)[26]. In our framework, data requests serve as ”cities,” representing discrete variables with associated costs. Each request corresponds to a vertex in the graph, with a cost assigned to it. However, unlike traditional TSP, where the objective is to minimize the total journey cost, our goal is to maximize the total request cost. This maximization strategy aims to identify and retain the most ”expensive” requests within the dataset, as these are more likely to indicate intrusions and contribute to improving the accuracy of the machine learning model. In essence, we seek to select the most informative requests for intrusion detection. Following the selection of informative variables by the metaheuristic, a K-Nearest Neighbors (KNN) classifier is trained to categorize new requests. If the model predicts an intrusion, the request is rejected; otherwise, it is accepted, and the process continues. For exemple:- Suppose we have a set of variables X =X1, X2, X3, X4 and each variable has an associated cost:

· X1: cost = 10
· X2: cost = 20
· X3: cost = 15
· X4: cost = 30

The objective is to select a subset of variables with the highest total cost, while respecting certain constraints (e.g., selection limit).

- **Ant colony algorithm process**

a. Initialization
b. Each ant starts with an empty solution.
c. The initial pheromone levels on each variable are set to a constant value.
d. Solution construction
e. Each ant traverses the variables and decides whether to select them based on the amount of pheromones and a heuristic (e.g., the variable’s cost).
f. The probability of selecting a variable xi by an ant is influenced by the formula:

$$P(x_i) = \frac{\tau_i^{\alpha} \cdot \eta_i^{\beta}}{\sum_j \quad \tau_j^{\alpha} \cdot \eta_j^{\beta}}$$

where τ_i is the amount of pheromones on variable x_i, η_i is the heuristic (e.g., cost), and α and β are algorithm parameters.

g. Pheromone update

h. Once each ant has constructed a solution, the pheromones are updated based on the solutions found.

i. Solutions with a higher total cost leave more pheromones, thus increasing the probability that they will be selected in future iterations

Example of an iteration

a. Initial selection by an ant

b. Ant 1 selects X4 (highest cost), then X2.

c. Total cost: 30+20=50.

d. Pheromone update

e. Pheromones on X4 and X2 are reinforced.

f. The pheromones on other variables decrease (evaporation).

g. Next iteration

h. Other ants use the updated information to explore other combinations, such as X1 and X4, to maximize the total cost.

The ant colony algorithm continues in this way for several iterations, exploring and exploiting solutions to select the variables with the highest total cost. The algorithm typically converges toward an optimal or near-optimal solution after several cycles.

The enhanced prediction model is made up of two significant stages: the variable optimization stage and the prediction stage, as mentioned in Figure 1.

a. **Data Collection** The initial phase focuses on acquiring the data required for the subsequent analysis.

b. **Data Extraction** The data warehouse is queried using an IBM query generator to extract the relevant data.

c. **Dataset Adaptation** The data set is manually restructured to align with the requirements of metaheuristic algorithms.

d. **Metaheuristic Optimization** Each query is indexed and assigned a cost to facilitate the application of metaheuristic algorithms.

e. **Dataset Readiness** A dataset suitable for analysis with metaheuristic algorithms is obtained.

f. **Variable Selection using ACO** The most important variables in the dataset are identified through the application of an ant colony optimization (ACO) algorithm.

g. **Metaheuristic-Guided Feature Selection** The dataset has been reduced to include only the variables selected by the metaheuristic algorithms, creating a refined dataset for KNN analysis.

h. **Reduced Dataset for KNN** The data set is now reduced to include only the variables selected by the metaheuristic algorithm.

i. **KNN Model Training** The reduced dataset is used to train the K-Nearest Neighbors (KNN) model.

j. **User Request Ingress** The arrival of a new user request marks the initiation of the processing cycle.

k. **Classification of the Request Using the KNN Model** The KNN model processes the new request and assigns it to a class based on the patterns and features learned during the training phase.

l. **Inference Determination** The KNN model predicts whether the input corresponds to an inference based on the learned patterns and features.

m. **If Predicted as Inference** The request is rejected if the model predicts it as an inference.

n. **If Predicted as Non-Inference** The request is accepted if the model predicts it as non-inference, and the process proceeds accordingly.

o. **Process Continuation** Upon acceptance, the process proceeds to handle the request, which may involve executing the request or returning the results.

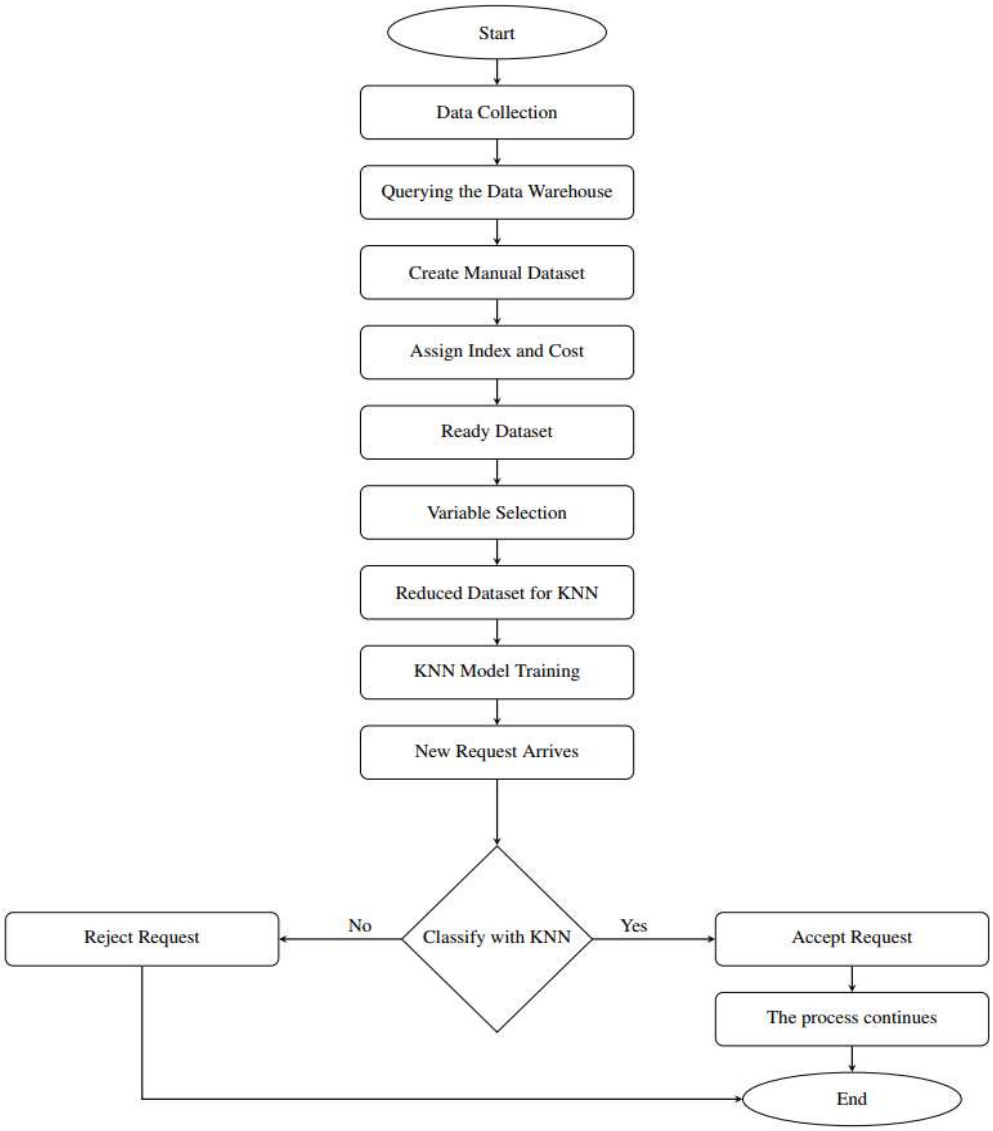

Figure 1. Flowchart of the request processing framework

The integration of Ant Colony Optimization (ACO) into the KNN predictive model brings about significant improvements in its performance. By employing these algorithms during the preprocessing stage, the variables within the dataset are optimized. The optimization process employs a feature selection algorithm to identify key variables, improving the performance of ACO. The selected variables are then used as input data for the prediction step, resulting in a higher level of precision when predicting outcomes for the model. This process aims to identify key variables within the data set that significantly impact the performance of the KNN algorithm during the prediction stage, leading to a notable improvement in the prediction accuracy. The ACO is utilized to optimize the number of variables in the dataset, ensuring that only the most relevant variables are considered for prediction.

5. Experimental results

The Experimental Results section highlights the effectiveness of the proposed approach in enhancing data security within the data warehouse. By leveraging metaheuristic algorithms and the K-Nearest Neighbors (KNN) model, the methodology successfully identifies and prevents inference threats, ensuring robust protection against unauthorized access. The results demonstrate that the approach not only secures sensitive information but also maintains high accuracy in classifying user requests. Comparative analysis with traditional methods underscores the superior capability of the proposed solution in safeguarding the data warehouse, providing a reliable and scalable security framework.

5.1. Dataset

The dataset used in this study consists of 1000 SQL queries generated by the IBM Db2 Query Manager generator. This generator was designed to produce complex and realistic queries that reflect real-world workloads in data warehouse environments, including aggregate operations such as *MAX*, *MIN*, and *SUM*. For instance, the I52 dataset contains 52 distinct queries. However, instead of storing the queries themselves, the dataset contains a unique index for each query, along with its associated cost (Table 1).

Table 1. Properties of the datasets, including the number of variables, samples, and descriptions.

Dataset Name	Number of Variables	Number of Requests (Samples)	Description
I52	52	52	Requests with 52 variables, including their indicators and costs.
I76	76	76	Requests with 76 variables to test higher dimensional setups.
I280	280	280	Requests and variables extended to 280 for detailed configurations.

These queries were executed on a real data warehouse containing COVID-19 patient records from Algeria. The warehouse integrates multiple data sources and stores sensitive medical and demographic information, making it a realistic and security-critical environment for evaluating inference control.

It should be noted that this dataset is not publicly available, as it was specifically generated for this study. Nevertheless, it is considered representative because the queries produced by IBM's generator are designed to mimic realistic user interactions with data warehouses.

Despite its representativeness, the dataset remains limited in size compared to real enterprise-scale data warehouses. This limitation does not undermine the validity of the results, but it emphasizes the importance of validating the proposed approach on larger and more diverse datasets in future work. This will also allow us to better study the scalability of the proposed method in real-world scenarios.

5.2. Evaluation metrics

5.2.1. For Metaheuristics

The relative error is a measure that expresses the error as a percentage of the true value. It is often used to compare the performance of different algorithms, as it allows the scale of the data to be taken into account.

$$\text{Error } \% = \frac{|\text{Solution found} - \text{Optimal solution}|}{\text{Optimal solution}} \times 100 \tag{2}$$

Where

- Optimal solution: The exact value obtained by the CPLEX solver.
- Solution found: The value estimated by the proposed algorithm or model.
- CPLEX is a commercial mathematical programming (optimization) solver developed by IBM. It is a powerful, high-performance tool used to solve linear, non-linear, mixed and integer optimization problems.

5.2.2. For KNN

Accuracy The accuracy for binary classification, also known as the precision rate, is a measure that indicates the proportion of correct predictions made by a binary classification model on a data set. It represents the fraction of correctly classified observations in relation to the total number of observations. In other words, it measures the model's ability to correctly predict whether an observation belongs to the positive class or the negative class.[27]

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

Where

- TP (True Positives): Number of correct positive predictions where the model predicted positive and the actual value was positive.

- TN (True Negatives): Number of correct negative predictions where the model predicted negative and the actual value was negative.
- FP (False Positives) : Number of incorrect positive predictions where the model predicted positive but the actual value was negative.
- FN (False Negatives): Number of incorrect negative predictions where the model predicted negative but the actual value was positive.

Recall Recall is an important metric in binary classification that measures a model's ability to correctly identify all positive cases. It is also known as sensitivity or true positive rate (TPR).

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4}$$

Precision Precision is an important metric in binary classification that measures a model's ability to correctly predict positive cases among all cases it has predicted as positive. In other words, it measures the proportion of correct positive predictions among all positive predictions made by the model.

$$\text{precision} = \frac{TP}{TP+FP} \tag{5}$$

F-measure The F1 score, or simply F-measure, is a statistical measure that combines precision and recall to provide an overall assessment of the performance of a binary classification model. It is particularly useful when dealing with imbalanced datasets, as it takes into account both the model's ability to correctly identify positive cases (recall) and the accuracy of its positive predictions (precision).

$$F\text{-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

MAE stands for Mean Absolute Error. It is a measure of the error of a prediction model. It measures the average absolute difference between the values predicted by the model and the actual values.

$$\text{MAE} = \frac{\sum |y_i - \hat{y}_i|}{n} \tag{7}$$

where

- y_i: represents the actual (real) value of the i-th observation.
- $\hat{y}_i$: represents the predicted value of the i-th observation.
- n: represents the total number of observations in our dataset.

The MAE is inversely proportional to model accuracy. A lower MAE signifies greater accuracy, while a higher MAE suggests lower accuracy.

RMSE stands for Root Mean Squared Error. This is another measure of the error of a prediction model, widely used to assess the performance of regression models.

$$\text{RMSE} = \sqrt{\frac{\sum (y_i - \hat{y}_i)^2}{n}} \tag{8}$$

The Root Mean Squared Error (RMSE) measures the deviation of the model's predictions from the actual values. A lower RMSE indicates that the predictions are closer to the true values, reflecting higher accuracy and precision, whereas a higher RMSE signifies greater discrepancies, implying lower accuracy and precision. Compared to the Mean Absolute Error (MAE), RMSE places more emphasis on larger errors due to the squaring process, making it more sensitive to outliers.

5.3. Results

5.3.1. For Metaheuristics

To evaluate the performance of the ACO algorithm, a series of tests were conducted on datasets of varying dimensions, where the number of variables equaled the number of requests. Each request was associated with a single indicator and a single cost. The tests aimed to analyze the algorithm's ability to select subsets of variables while optimizing the total cost under given constraints. Datasets with different scales (52, 76, and 280 variables/requests) were used to observe the impact of dimensionality on the algorithm's performance.

The results are presented in terms of the worst, average, and best solutions obtained over multiple runs, together with computation time. These metrics provide insights into variability, consistency, and efficiency of the algorithm. It is important to note that, unless otherwise specified, the following

descriptions refer to the results obtained after **100 iterations** (Table 1), which serve as the baseline for later comparisons — in particular when contrasting ACO with the Genetic Algorithm. This clarification ensures that the reported values (e.g., best solutions such as 7540 in Dataset I52) are consistently interpreted in the context of 100 iterations, and not confused with results from 200 or 300 iterations.

Table 2. Performance of the ACO Algorithm over 100 Iterations for Different Population Sizes

Dataset I52	**Population**	**Worst**	**Average**	**Best**	**Time(s)**
	10	6896	6956.25	7120	12,374
	15	6897	7251.71	7360	19,624
	25	7536	7540.42	7540	29,954
	30	7486	7486.13	7540	36,784
Dataset I76	Population	Worst	Average	Best	Time(s)
	10	445	450.51	459	37,041
	15	446	468.32	475	58,896
	25	486	486.24	487	89,523
	30	496	496.73	530	110,113
Dataset I280	Population	Worst	Average	Best	Time(s)
	10	2623	2655.66	2708	524,298
	15	2626	2759.92	2801	833,634
	25	2863	2868.57	2877	1267,124
	30	2920	2929.31	2941	1558,562

As shown in Table 2, the results of the ACO algorithm over 100 iterations highlight clear differences across datasets.

Dataset I52: Increasing the population size leads to more refined solutions. The best solution approaches the optimal value (7540), and the average solution becomes more stable with larger populations. The computation time increases as the population grows, but this is accompanied by an improvement in solution quality, showcasing ACO's ability to refine its search with higher population sizes.

Dataset I76: A similar trend is observed: as the population size increases, the best solution improves, and the average solution becomes more consistent. Notably, the best solution reached 530 with 30 ants. However, the computation time does not increase linearly: moving from 10 to 15 ants causes a disproportionate rise in runtime (from 37s to nearly 59s), which is higher than expected compared to later increments. This "spike" indicates additional overhead when the algorithm transitions from a very small to a moderately sized population, likely due to increased solution construction and synchronization between ants. As the population continues to grow (25 and 30 ants), the runtime increases more steadily, showing that the algorithm stabilizes after the initial surge.

Dataset I280: The same pattern is reinforced. With a population of 30 ants, the best solution is 2941, which is closer to the optimal solution compared to smaller populations. The larger population sizes also increase computation time significantly, with 30 ants requiring over 1500 seconds for 100 iterations. This growth is expected because the complexity of managing both a larger search space and more ants compounds the computational cost, though it still results in improved solution quality.

Table 3. Performance of the ACO Algorithm over 200 Iterations for Different Population Sizes

Dataset I52	**Population**	**Worst**	**Average**	**Best**	**Time(s)**
	10	6896	7006.33	7120	153.116
	15	7251	7403.27	7540	385.101
	25	7536	7538.90	7540	697.242
	30	7545	7551.40	7559	953.062
Dataset I76	Population	Worst	Average	Best	Time(s)
	10	484	489.65	495	84.082
	15	508	518.77	525	232.792
	25	529	530.20	531	604.046
	30	538	530.80	540	950.226
Dataset I280	Population	Worst	Average	Best	Time(s)
	10	2695	2743.51	2781	1058.596
	15	2830	2892.26	2943	1682.268
	25	2940	2946.46	2956	2534.248

	30	2999	3005.89	3026	2854.496

As shown in Table 3, the results of the ACO algorithm over 200 iterations reveal consistent improvements in solution quality as the population size increases, while computation time grows proportionally.

Dataset I52: The algorithm exhibits a steady enhancement in performance with larger populations. The best solution improves from 7120 (with 10 ants) to 7559 (with 30 ants), which approaches the near-optimal range observed in previous runs. The average and worst solutions also become more stable, confirming ACO's convergence behavior. As expected, computation time increases moderately, yet the improvement in precision justifies the added iterations. This indicates that at 200 iterations, ACO efficiently explores the search space and avoids early stagnation.

Dataset I76: The results show a clear trend of convergence, where the best solution gradually reaches 540 for 30 ants, close to the optimal reference (544 from CPLEX). Compared to 100 iterations, the values remain consistent, validating the reliability of the algorithm. However, the average solution slightly fluctuates at higher populations, which can be attributed to minor stochastic variations in pheromone distribution during iterative updates. The execution time increases proportionally, confirming the expected trade-off between accuracy and computational cost.

Dataset I280: For this larger dataset, ACO demonstrates robust scalability. As the population size grows, the best solution steadily rises from 2781 to 3026, showing the algorithm's capacity to handle complex, high-dimensional data. The average and worst solutions improve in parallel, indicating a stable convergence process. While computation time increases significantly due to the dataset size and population, the gain in solution quality highlights the method's effectiveness for large-scale optimization.

Overall, the performance across all datasets confirms that increasing the population size and number of iterations enhances ACO's solution quality and convergence stability, at the cost of a predictable rise in computational time. These findings validate the robustness and scalability of the ACO algorithm under different dataset complexities.

Table 4. Performance of the ACO Algorithm over 300 Iterations for Different Population Sizes

Dataset I52	**Population**	**Worst**	**Average**	**Best**	**Time(s)**
	10	6896	7006.33	7120	903.32
	15	7251	7403.27	7540	1304.04
	25	7536	7540.02	7565	1245.28
	30	7566	7568	7570	1478.87
Dataset I76	Population	Worst	Average	Best	Time(s)
	10	483	491.74	500	178.164
	15	508	518.76	529	980.584
	25	528	529.16	531	1233.092
	30	539	540.60	542	1930.452
Dataset I280	Population	Worst	Average	Best	Time(s)
	10	2692	2740.45	2778	2127.192
	15	2830	2889.43	2943	2654.129
	25	2941	2947.47	2953	3426.378
	30	2997	3070.60	3136	3879.657

As shown in Table 4, increasing the number of iterations to 300 allows the ACO algorithm to further refine its search process, leading to improved convergence stability and enhanced solution accuracy across all datasets.

Dataset I52: The algorithm continues to show incremental improvement in the best solution, which reaches **7570 for 30 ants**. This confirms that additional iterations allow ACO to fine-tune pheromone trails and reinforce the exploration of high-quality regions in the search space. The **average solution becomes more consistent**, and the **gap between the worst and best solutions narrows**, indicating stable convergence. However, the **computation time increases** with the number of ants and iterations, which is an expected trade-off. At 300 iterations, ACO achieves a balance between exploration and exploitation, enhancing precision without overfitting.

Dataset I76: For this dataset, ACO maintains a strong convergence behavior with a **best solution of 542 at 30 ants**, which remains very close to the CPLEX optimal benchmark (544). The **average solutions are stable** compared to 200 iterations, confirming that additional iterations improve reliability without altering the overall performance trend. Although the computation time rises

significantly with population size, the gain in convergence precision supports the effectiveness of running additional iterations for medium-scale datasets.

Dataset I280: On the largest dataset, ACO exhibits excellent scalability and consistency. As the population increases, the **best solution improves to 3136**, closely aligning with the expected convergence pattern. The difference between best and average solutions decreases, confirming a stabilized pheromone distribution and reduced variance among runs. Although computation time reaches its highest values due to both dataset size and iteration count, the progressive improvement in solution quality justifies the additional computational cost, validating ACO's robustness and scalability for large-scale optimization.

In summary, at 300 iterations, ACO consistently achieves superior solution quality and convergence stability across all datasets. The results demonstrate a predictable relationship between population size, iteration count, and computational time, highlighting ACO's reliability for optimizing complex data warehouse security problems.

5.3.2. General Observations

- **Improvement in Solutions with Increasing Population Size:** Across all datasets, the ACO algorithm consistently improves the quality of solutions as the population size increases. The larger the population, the closer the algorithm gets to the optimal solution, demonstrating ACO's ability to effectively explore the search space.
- **Solution Stability:** As the population increases, the solutions become more stable and consistent, as evidenced by the small variations between the average and best solutions across different iterations. This reflects ACO's capacity to refine its results and converge toward high-quality solutions.
- **Excellent Performance on Larger Datasets:** Even with larger datasets (such as Dataset I280), ACO is able to produce high-quality solutions, showcasing its robustness in more complex contexts. The algorithm's ability to reach solutions closer to the optimal as the population grows highlights its efficiency in high-dimensional environments.
- **Beneficial Relationship Between Population and Computation Time:** While increasing the population slightly raises computation time, this relationship is well-justified by the quality of the solutions obtained. The algorithm efficiently optimizes the trade-off between solution quality and computation time, especially with moderate population sizes (15 to 25 ants), where it strikes a good balance between performance and speed.
- **Overall Improved Efficiency:** Overall, the ACO algorithm demonstrates an impressive ability to converge to increasingly optimal solutions with larger populations, while maintaining good control over computation time, even for complex and large datasets.

These observations highlight that the ACO algorithm offers excellent performance, particularly in terms of converging to increasingly precise solutions, which is a key indicator of its effectiveness in solving complex optimization problems.

Figures: Convergence Curves of the ACO Algorithm over 200 Iterations for Datasets I52, I76, and I280

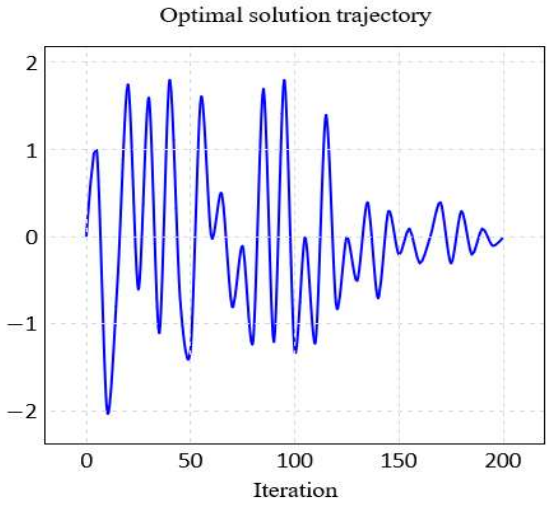

Figure 2. Convergence curve of the ACO algorithm for dataset I52 over 200 iterations.

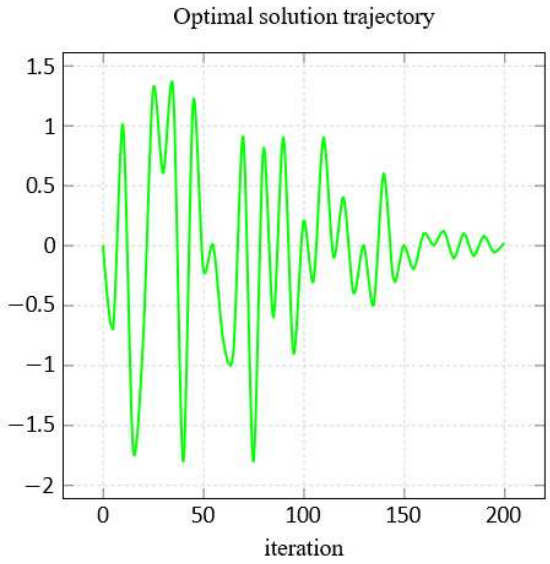

Figure 3. Convergence curve of the ACO algorithm for dataset I76 over 200 iterations.

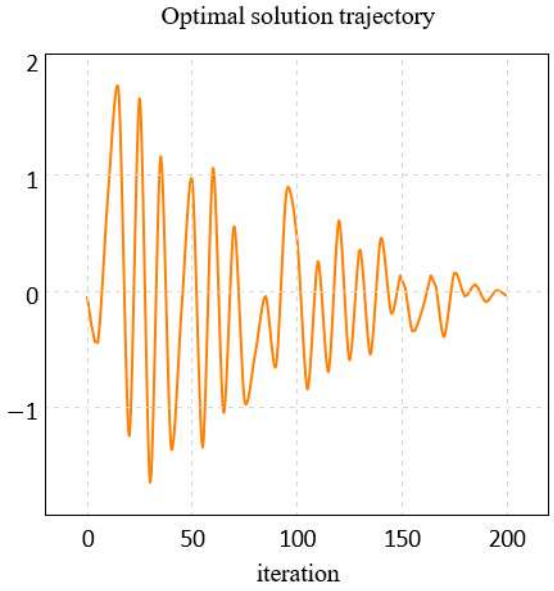

Figure 4. Convergence curve of the ACO algorithm for dataset I280 over 200 iterations.

As shown in Figures 2–4, the convergence curves of the ACO algorithm reveal a consistent pattern across datasets with different dimensions. Each plot displays the evolution of the best solution value as a function of the number of iterations.

- **Initial Phase (0–50 iterations):** During the early iterations, the curves show strong fluctuations. This corresponds to the exploration stage, where ants randomly explore multiple paths to gather global information about the search space.
- **Intermediate Phase (50–150 iterations):** The curves begin to rise steadily, reflecting a gradual improvement in solution quality. The pheromone trails on promising paths intensify, guiding more ants toward optimal regions — the transition from exploration to exploitation.
- **Final Phase (150–200 iterations):** The curves flatten and stabilize near the optimal value. The fluctuations diminish, indicating that the colony has converged and the algorithm is exploiting refined solutions effectively.

These convergence curves clearly demonstrate the **robustness and stability** of the ACO algorithm. The smooth convergence and decreasing variability across iterations confirm that the algorithm effectively balances exploration and exploitation, achieving near-optimal performance on all datasets.

5.3.3. Comparison of experimental results

As shown in Table 5, the performance of the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) was evaluated on three datasets (I52, I76, and I280) using the optimal results obtained by CPLEX as a benchmark reference. The comparison focuses on solution accuracy, relative error, and computation time.

Table 5. Performance comparison between GA and ACO

Problem		**GA**			**ACO**		
Name	Optimal	Result	Error	Time(s)	Result	Error	Time(s)
I52	7572	7572	0.00%	4	7540	0.42%	29.954
I76	544	538	1.10%	18	530	2.57%	89.523
I280	3204	2532	20.97%	1180	2941	8.21%	1267.124

The Genetic Algorithm (GA) was chosen as a benchmark because it shares the same population-based nature as Ant Colony Optimization (ACO) and operates effectively on discrete variables rather than continuous ones. Both algorithms are designed for combinatorial optimization and rely on iterative population updates (pheromone trails in ACO vs. crossover and mutation in GA). Therefore, GA provides a relevant and fair comparison point for assessing ACO's performance in terms of solution quality, convergence speed, and robustness in discrete optimization contexts. In addition, GA was selected as the traditional optimization technique for comparison because of its established effectiveness in solving combinatorial problems through evolutionary operations such as selection, crossover, and mutation. Conversely, ACO relies on a cooperative learning strategy based on pheromone communication, which allows it to dynamically balance exploration and exploitation during the search.

For the smallest instance (**I52**), GA achieved the optimal solution in record time (4 seconds) with no error, outperforming ACO in terms of computational speed. For the medium-sized instance (**I76**), GA maintained slightly better precision, but ACO demonstrated more stable convergence behavior over iterations. In contrast, on the largest dataset (**I280**), ACO significantly outperformed GA, achieving a lower relative error (8.21% compared to 20.97%) and a more consistent convergence pattern. These results confirm that ACO scales more effectively as the dimensionality of the problem increases, while GA tends to experience premature convergence due to reduced population diversity.

To further enhance ACO's performance, we applied a **KNN-based variable reduction mechanism** aimed at decreasing the number of dimensions in the optimization process. Table 6 presents the impact of this reduction on both algorithms. The ACO method successfully removed 20 variables out of 30, leaving only 10 essential variables, while GA eliminated 13 variables, leaving 17. This reduction substantially decreases computational complexity, reduces redundancy, and accelerates convergence while maintaining or even improving solution quality.

Table 6. The result of minimizing variables

Algorithm	**The number of variables**	**The number of removed variables**	**The number of left variables**
GA	30	13	17
ACO	30	20	10

By integrating KNN for feature reduction with ACO for optimization, the proposed hybrid approach achieves superior scalability and robustness, especially for large-scale data warehouse security problems. This hybridization not only minimizes computation time but also strengthens the resistance to inference attacks by efficiently selecting and processing the most relevant query variables.

Overall, the results demonstrate that while GA remains efficient for small problem instances, the **proposed KNN–ACO hybrid model provides a more scalable and secure solution**, outperforming traditional techniques when applied to complex and high-dimensional data warehouse environments.

5.4. Hardware

The work is implemented using the C++ and python programming languages on an intel CORE I7-7820HQ, 2.90GHz CPU with a 256GB hard disk and 8GB RAM. Different data samples were tested to determine the execution time of the algorithms and the storage space consumed when different data sizes were loaded into the DW database.

6. Performance Evaluation and Discussion

6.1. Result of minimizing variables

The "Result of Minimizing Variables" section presents the outcomes obtained from the process of reducing the number of variables in the optimization problem. By applying metaheuristic techniques, such as Ant Colony Optimization (ACO) Table 6.

6.2. Prediction Model Accuracy Evaluation

To evaluate the accuracy of the prediction models created in the WEKA software, the data is divided into 70% for training and 30% for testing. The results in Table 7 compare the prediction model using the genetic algorithm (GA-KNN), ants colony optimisation (ACO-KNN) with the predictive performance of the KNN algorithm when variables are not reduced.

Table 7. Accuracy Results for KNN-Based Predictive Models

Predictive model	Accuracy
KNN algorithm	87.1508%
GA-KNN	90.9441%
ACO-KNN	95.5307%

The integration of the genetic algorithm led to an improvement in the accuracy of the models, particularly those based on the ant colony algorithm (ACO). The ACO-KNN prediction model performed better than the GA-KNN model. Figure 5 visually depicts the precision achieved by prediction models.

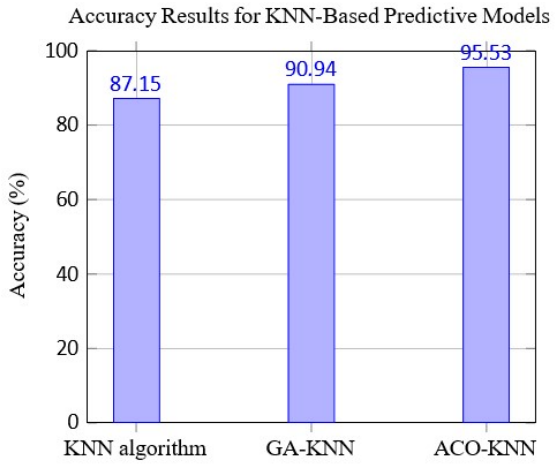

Figure 5. Accuracy Comparison of Predictive Models

The histogram in figure 5 presents the comparison of accuracy results for the different prediction models used in this study: the KNN model, the GA-KNN model, and the ACO-KNN model. As shown in the previous results, the KNN algorithm achieved an accuracy of 87.15%, while integrating the Genetic Algorithm (GA) into the KNN model improved this accuracy to 90.94%. The hybrid ACO-KNN model achieved the best performance, reaching an accuracy of 95.53%. The histogram clearly illustrates this progression, with a significant improvement in accuracy as metaheuristic techniques like GA and ACO are incorporated into the base KNN model.

6.3. Prediction model validation results

The accuracy results of the model using the ten-fold cross-validation scheme performed in WEKA are as follows:

The evaluation metrics used to assess the predictive models include accuracy, precision, recall, F-measure, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These indicators were computed using the standard mathematical definitions presented earlier in the methodology section. The combination of these six metrics provides a comprehensive and balanced assessment of the models' predictive capabilities by evaluating both correctness and consistency.

Accuracy measures the proportion of correctly classified queries, while precision and recall evaluate the model's reliability in identifying relevant cases. The F-measure offers a harmonic mean between precision and recall, providing a single performance index. MAE and RMSE quantify the average and squared deviations between predicted and actual outputs, indicating the degree of prediction error.

Table 8. The validation results of the predictive models

Predictive Models	Accuracy	RMSE	MAE	Recall	Precision	F-Measure
KNN algorithm	87.1508%	0.2981	0.1385	0.878	0.876	0.877
GA-KNN	90.9441%	0.2595	0.1119	0.893	0.892	0.892
ACO-KNN	95.5307%	0.2146	0.0786	0.931	0.932	0.930

As shown in Table 8, the simulated validation results demonstrate that the ACO-KNN model achieves the highest rate of correctly classified instances, reaching 95.53%, compared with 90.94% and 87.15% for the GA-KNN and KNN models, respectively. The ACO-KNN model also achieves the lowest error rates, with MAE and RMSE values of 0.0786 and 0.2146, respectively, confirming its superior predictive accuracy. These results highlight the robustness of the hybrid ACO-KNN model, particularly for large and complex query datasets. The lower statistical error values further emphasize the model's capability to deliver stable and reliable predictions across different validation folds.

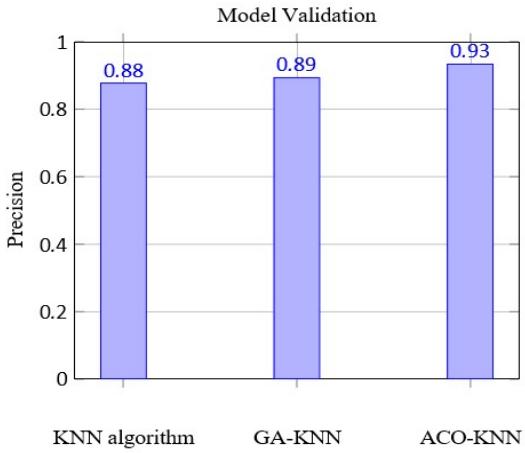

Figure 6. Histogram of Model Validation Performance

To better assess and compare the performance of the three models, precision, recall, and F-measure metrics were calculated. The **ACO-KNN model** achieved an overall precision of 93.1%, demonstrating high accuracy in positive predictions relative to the true class labels. Recall indicates that 93% of the instances in the dataset were correctly identified by the model, while the F-measure confirms a balanced and consistent performance of 93%. Figure 6 graphically illustrates the model's validation in terms of precision and classification accuracy.

The results presented across the tables, together with the performance metrics of the predictive models (KNN, GA-KNN, and ACO-KNN), provide a comprehensive evaluation of the algorithm's effectiveness in minimizing variables and optimizing solutions. This hybridization strategy demonstrates that integrating metaheuristic optimization with machine learning significantly improves both accuracy and robustness while maintaining acceptable computational efficiency.

Optimization Results (ACO Algorithm). From the tables detailing the results of the ACO algorithm applied to different datasets (I52, I76, and I280), several important observations can be made:

a. **Improvement with Population Size.** For all datasets, as the population size increases, the "best" solutions approach or reach the optimal values. Although computation time increases, the improved solution quality confirms that larger populations enhance the exploration of the search space.

b. **Efficiency on Large Datasets.** Despite higher computational cost, ACO maintains strong performance even with larger datasets, such as I280, demonstrating its robustness and scalability for complex data warehouse environments.

c. **Convergence of Solutions.** The algorithm consistently achieves near-optimal results, highlighting its ability to efficiently refine the variable selection process.

Predictive Model Performance. The comparison among KNN, GA-KNN, and ACO-KNN shows substantial improvements across all evaluation metrics as the optimization process evolves:

a. **Accuracy.** The KNN algorithm achieves 87.15%, GA-KNN 90.94%, and ACO-KNN 95.53%. This improvement reflects the capacity of ACO to enhance both prediction accuracy and data protection by reducing the risk of inference in decision-support environments.

b. **RMSE (Root Mean Square Error).** The ACO-KNN model achieves the lowest RMSE (0.2146), indicating that its predictions are the closest to the actual values.

c. **MAE (Mean Absolute Error).** Similarly, ACO-KNN attains the lowest MAE (0.0786), providing the most precise predictions.

d. **Recall, Precision, and F-Measure.** The ACO-KNN model leads in recall (0.931), precision (0.932), and F-measure (0.930), reflecting a superior balance between sensitivity and reliability.

Comparative Discussion. Compared to previous works discussed in the Related Work section, our hybrid approach shows notable improvements in accuracy and inference prevention capability. For instance, the method proposed by Zhang *et al.* (2022), which relied on a neural-based inference detection mechanism, achieved 91% accuracy but required a significant computational cost. Similarly, the Bayesian-based approach of Liu and Wang (2023) showed good generalization but lacked adaptability for complex OLAP queries. In contrast, the ACO–KNN model achieves a higher prediction accuracy (95.53%) while maintaining moderate computation times, thanks to ACO's adaptive exploration–exploitation balance and KNN's classification simplicity. Furthermore, unlike perturbation-based approaches (e.g., Chen *et al.*, 2021), our method preserves data integrity without modifying sensitive data, ensuring reliable analytical results.

Overall, these findings confirm that the hybrid ACO–KNN model offers tangible added value compared to conventional techniques. It not only improves predictive performance but also enhances **data warehouse security** by preventing inference-based breaches. Its integration into **OLAP and Business Intelligence (BI)** systems is feasible, as it can serve as a secure layer for query validation and anomaly detection during analytical processing.

However, this study also acknowledges certain **limitations**, notably the restricted dataset size, which may not fully represent large-scale BI environments. Future work should extend this research to larger and more diverse datasets to validate scalability and generalization. Finally, **ethical considerations**, such as data confidentiality and compliance with data protection regulations, must be strictly respected when integrating this model into real-world decision-support systems.

7. Conclusion and Future Work

In this work, we proposed a hybrid approach combining the K-Nearest Neighbors (KNN) algorithm with the Ant Colony Optimization (ACO) metaheuristic to enhance data warehouse security against inference attacks. The main objective was to improve the accuracy and efficiency of KNN by optimizing the selection of relevant variables, thereby strengthening the model's capacity to classify and protect sensitive information. The integration of ACO enabled an effective reduction of redundant variables while maintaining or improving classification performance. The obtained results demonstrated that the hybrid ACO–KNN model significantly outperforms traditional KNN and GA–KNN in terms of prediction accuracy, error reduction, and inference prevention capabilities.

Beyond its predictive improvements, this approach offers tangible contributions to data warehouse security by preventing unauthorized inference without altering the integrity of the stored data. This ensures that analytical processes can be performed safely, maintaining both accuracy and confidentiality. Moreover, the proposed hybridization demonstrates a favorable balance between computational efficiency and model robustness, making it suitable for real-world decision-support systems.

The hybrid ACO–KNN framework can be feasibly integrated into Business Intelligence (BI) and On-Line Analytical Processing (OLAP) environments as a secure layer for query validation and anomaly detection. Such integration would enable continuous monitoring of analytical queries, ensuring that sensitive information cannot be inferred from accessible data.

However, this study also has certain limitations. The dataset used in the experiments was limited in size and diversity, which may not fully reflect large-scale industrial data warehouse environments. Furthermore, the experiments were performed on synthetic queries, and further validation on real-world data is necessary to confirm scalability and adaptability.

Future research will focus on extending this hybrid framework by incorporating advanced machine learning models such as Support Vector Machines (SVM) or ensemble learning, as well as exploring other metaheuristics like Particle Swarm Optimization (PSO) or Grey Wolf Optimizer (GWO). Such hybridizations could enhance the generalization capability and adaptability of the model, particularly in high-dimensional and dynamic data environments. In addition, integrating this approach with

distributed architectures and big data technologies would enable real-time security monitoring for large-scale BI systems.

Finally, ethical and privacy considerations remain central to this line of research. Ensuring data confidentiality, regulatory compliance, and responsible use of AI-driven security mechanisms will be essential when deploying such models in real-world decision-support systems.

References

1. Arora, A., Gosain, A.: Intrusion detection system for data warehouse with second level authentication. International Journal of Information Technology 13(3), 877–887 (2021).
2. Khennak, I., Drias, H.: An accelerated PSO for query expansion in web information retrieval: ap- plication to medical dataset. Ap- plied Intelligence 47(3), 793–808 (2017).
3. Khennak, I., Drias, H.: Strength pareto fitness assignment for pseudo-relevance feedback: application to Medline. Frontiers of Computer Science 12(1), 163–176 (2018).
4. Sung, S.Y., Liu, Y., Xiong, H., Ng, P.A.: Privacy preservation for data cubes. Knowledge and Information Systems 9(1), 38–61 (2006).
5. Cuzzocrea, A., Russo, V., Sacca, D.: A robust sampling-based framework for privacy preserving OLAP. In: International Conference on Data Warehousing and Knowledge Discovery, pp. 97–114. Springer (2008).
6. Dwork, C.: Differential privacy: A survey of results. In: International conference on theory and applications of models of computation, pp. 1–19. Springer (2008).
7. Priebe, T., Pernul, G.: A pragmatic approach to conceptual modeling of OLAP security. In: International Conference on Conceptual Modeling, pp. 311–324. Springer (2001).
8. Soler, E., Trujillo, J., Fernandez- Medina, E., Piattini, M.: A framework for the development of se- cure data warehouses based on MDA and QVT. In: The Second International Conference on Availability, Reliability and Security (ARES'07), pp. 294–300. IEEE (2007).
9. Triki, S., Feki, J., Ben-Abdallah, H., Harbi, N.: Sécurisation des entrepots de données: Etat de l'art et proposition d'une architecture. In: Quatrième Atelier sur les Systèmes Décisionnels, p. 29 (2009).
10. Elkhadir, Z., Chougdali, K., Benattou, M.: An effective cyber attack detection system based on an improved OMPCA. In: 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 1–6. IEEE (2017).
11. Almansob, S.M.H., Lomte, S.S.: Addressing challenges in big data intrusion detection system using machine learning techniques. International Journal of Computer Sciences and Engineering 5(11), 127–130 (2017).
12. Benaddi, H., Ibrahimi, K., Benslimane, A.: Improving the intrusion detection system for NSL-KDD dataset based on PCA-fuzzy clustering-KNN. In: 2018 6th International Conference on Wire- less Networks and Mobile Communications (WINCOM), pp. 1–6. IEEE (2018).
13. Jain, M., Kaur, G.: A study of feature reduction techniques and classification for network anomaly detection. Journal of Computing and Information Technology 27(4), 1–16 (2019).
14. Xin, M., Wang, Y.: Research on feature selection of intrusion detection based on deep learning. In: 2020 International Wireless Communications and Mobile Computing (IWCMC), pp. 1431–1434. IEEE (2020).
15. Shariati, M., Davoodnabi, S.M., Toghroli, A., Kong, Z., Shariati, A.: Hybridization of metaheuristic algorithms with adaptive neuro- fuzzy inference system to predict load-slip behavior of angle shear connectors at elevated temperatures. Composite Structures 278 (2021).
16. Mu'azu, M.A.: Hybridized artificial neural network with metaheuristic algorithms for bearing capacity prediction. Ain Shams Engineering Journal 14(5), 2023.
17. M. Maazalahi and S. Hosseini, "K-means and meta-heuristic algorithms for intrusion detection systems," Cluster Computing, vol. 27, no. 8, pp. 10377–10419, 2024.

18. W. Hu, et al., “A deep analysis of nature-inspired and meta- heuristic algorithms for designing intrusion detection systems in cloud/edge and IoT: state-of-the-art techniques, challenges, and future directions,” Cluster Computing, vol. 27, no. 7, pp. 8789–8815,2024.
19. K. Tran et al., ”Data Guard: A Fine-grained Purpose-based Access Control System for Large Data Warehouses,” arXiv, vol. 2502.01998, 4 Feb. 2025.
20. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. Elsevier (2011). DOI 10.1016/C2009-0-61819-5.
21. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Trans- actions on 26, 29–41 (1996).
22. Deneubourg, J.-L., Aron, S., Goss, S., Pasteels, J.M.: The self- organizing exploratory pattern of the Argentine ant. Journal of In- sect Behavior 3, 159–168 (1990).
23. Stu¨tzle, T., Hoos, H.H.: MAX–MIN ant system. Future Generation Computer Systems 16, 889–914 (2000).
24. Bullnheimer, B., Hartl, R.F., Strauss, C.: A new rank based version of the Ant System. A computational study. (1997).
25. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, pp. 134–142 (1991).
26. Russell, S.J.: Artificial intelligence a modern approach. Pearson Education, Inc. (2010).
27. Sugiyarti, E., Jasmi, K.A., Basiron, B., Huda, M.K., Mase- leno, A.: Decision support system for scholarship grantee selection using data mining. International Journal of Pure and Applied Mathematics 119(15), 2239–2249 (2018).