KMAN PUBLISHING INSTITUTE

Article history:
Received 19 December 2025
Revised 17 April 2026
Accepted 20 April 2026
Published online 01 July 2026

Journal of Resource Management and Decision Engineering

Volume 5, Issue 4, pp 1-27

An Enhanced Hybrid Statistical–Learning Approach for Detecting Fake Feedback in Cloud Environments Using Hierarchical Clustering with Reptile Search Algorithm and Particle Swarm Optimized Extreme Learning Machine

Hussein Kadhim. Almamoori[1], Golnaz. Aghaee Ghazvini[2*], Ali. Albu-Rghaif[3], Fariba. Majidi[1]

[1] Department of Computer Engineering, Isf.C., Islamic Azad University, Isfahan, Iran
[2] Department of Computer, Dol.C., Islamic Azad University, Isfahan, Iran
[3] Department of Computer Engineering, Diyala Branch, Diyala University, Diyala, Iraq

*** Corresponding author email address**: g.aghaee@iau.ac.ir

A r t i c l e I n f o

Article type:
Original Research

How to cite this article:
Almamoori, H. K.., Aghaee Ghazvini, G., Albu-Rghaif, A., & Majidi, F. (2026). An Enhanced Hybrid Statistical–Learning Approach for Detecting Fake Feedback in Cloud Environments Using Hierarchical Clustering with Reptile Search Algorithm and Particle Swarm Optimized Extreme Learning Machine. *Journal of Resource Management and Decision Engineering*, *5*(4), 1-27.
https://doi.org/10.61838/kman.jrmde.285

A B S T R A C T

The proliferation of fake feedback in cloud environments represents a major challenge in maintaining user trust and ensuring service quality. Accordingly, the development of intelligent approaches for distinguishing genuine feedback from fraudulent instances has become increasingly important. In this study, a hybrid framework based on statistical analysis and machine learning is proposed for the detection of fake feedback. The framework integrates improved hierarchical clustering using the Reptile Search Algorithm (AHC-RSA), the E-EDA statistical model for anomaly filtering, and an Extreme Learning Machine optimized with Particle Swarm Optimization (ELM-PSO). This integrated approach reduces intra-cluster dispersion, enhances data separability, improves prediction accuracy, and accelerates model convergence. Experimental evaluations conducted on the CloudArmor and Epinions datasets demonstrated that the proposed model achieved accuracies of 99.83% and 99.84%, respectively. The results indicate that the proposed model outperforms SVM, ANN, LSTM, and GRU algorithms in terms of accuracy, stability, and convergence speed. On average, the proposed model shows an improvement of approximately 5% to 8% in accuracy and F1-score, along with a significant reduction in MSE and RMSE compared to the best existing methods. Therefore, the proposed framework can be considered an effective and reliable approach for detecting fake feedback and enhancing trust in cloud services.

Keywords: *Fake feedback detection, clustering with Reptile Search Algorithm, enhanced empirical analysis method, Particle Swarm Optimization-based Extreme Learning Machine*

1. Introduction

The rapid expansion of cloud computing has fundamentally transformed the way organizations store, process, and exchange data, enabling scalable, flexible, and cost-efficient service delivery across diverse application domains. Cloud environments provide on-demand access to computational resources and services, thereby facilitating digital transformation and enhancing operational efficiency for both enterprises and individual users (Guo et al., 2023; Islam et al., 2023). However, alongside these benefits, the increasing reliance on cloud-based systems has introduced significant security and trust-related challenges, particularly in the context of user-generated feedback and service evaluation mechanisms. In such environments, feedback provided by users plays a crucial role in guiding decision-making processes, influencing service selection, and establishing trust among cloud service providers and consumers (Aghaee Ghazvini et al., 2020; Mujawar & Bhajantri, 2022). Consequently, ensuring the authenticity and reliability of feedback has become a critical concern in maintaining the integrity of cloud ecosystems.

One of the most pressing issues in this domain is the proliferation of fake feedback, also referred to as opinion spam or fraudulent reviews. Fake feedback is intentionally generated to manipulate perceptions, distort trust metrics, and influence user behavior, often for malicious or competitive advantage. The presence of such deceptive information undermines the credibility of cloud services and leads to suboptimal decision-making by users (Ahmed et al., 2018; Salminen et al., 2022). Moreover, fake feedback can significantly distort trust management systems, which rely heavily on user evaluations to assess the quality and reliability of cloud service providers (Siadat et al., 2017; Soleymani et al., 2021). As a result, the detection and elimination of fake feedback have become essential for ensuring secure and trustworthy cloud environments.

Traditional approaches to trust evaluation in cloud systems have primarily focused on rule-based and statistical methods. Early models, such as trust computation frameworks and fuzzy rule-based systems, attempted to evaluate the credibility of feedback by analyzing user behavior and historical interactions (Jagpreet & Sarbjeet, 2017; Shilpa & Rajesh, 2018). While these methods provided initial insights into trust assessment, they often lacked the capability to effectively handle large-scale, high-dimensional, and dynamic datasets characteristic of modern cloud environments. Furthermore, such approaches are typically limited in their ability to detect sophisticated patterns of fraudulent behavior, particularly when attackers employ coordinated or adaptive strategies (Wang et al., 2020; Ye & Akoglu, 2015).

With the advancement of artificial intelligence and machine learning, more sophisticated techniques have been introduced to address the challenges of fake feedback detection. Supervised learning algorithms, including Decision Trees, Support Vector Machines (SVM), and k-Nearest Neighbors (KNN), have been widely used for classification tasks in this domain (Ram et al., 2022; Thirunavukkarasu et al., 2023). These methods leverage labeled datasets to learn patterns that distinguish genuine feedback from fraudulent instances. However, their performance is often constrained by the quality and availability of labeled data, as well as their limited ability to capture complex nonlinear relationships within the data. In addition, many real-world datasets suffer from class imbalance, where fake feedback instances are significantly fewer than genuine ones, leading to biased model performance (Alsubari et al., 2022; Asaad et al., 2023).

Recent research has explored the application of deep learning techniques to enhance fake feedback detection. Models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) have demonstrated improved performance by capturing semantic, temporal, and contextual features of textual feedback (Deshai & Rao, 2023; Javed et al., 2021). These models are particularly effective in processing large-scale textual data and identifying subtle linguistic patterns associated with deceptive reviews. However, deep learning approaches often require substantial computational resources and large annotated datasets, which may not always be available in cloud-based applications. Additionally, these models may suffer from overfitting and lack interpretability, limiting their practical deployment in real-world scenarios (Hajek et al., 2020; Manaskasemsak et al., 2023).

Another promising direction in fake feedback detection involves the use of data analytics and anomaly detection techniques. Methods based on statistical analysis, density estimation, and clustering have been employed to identify outliers and unusual patterns in feedback data (Gu, 2017; Shi et al., 2020). For instance, hierarchical clustering methods have been used to group similar feedback instances and detect anomalies within clusters, thereby improving detection accuracy (Shi et al., 2020). Similarly, topic

modeling and network-based approaches have been applied to uncover hidden relationships among users and identify coordinated fraudulent activities (Birim et al., 2022; He et al., 2022). Despite their effectiveness, these approaches often face challenges related to scalability, sensitivity to parameter selection, and the inability to fully exploit both structural and semantic information in the data.

Hybrid models that integrate multiple techniques have recently gained attention as a means of overcoming the limitations of individual methods. By combining clustering, statistical analysis, and machine learning, these models aim to leverage the strengths of each approach to achieve more robust and accurate detection performance (Cho et al., 2023; Truong et al., 2023). For example, ensemble learning models and hybrid frameworks have been proposed to improve classification accuracy and reduce false detection rates in fake review identification (Barbado et al., 2019; Taneja & Kaur, 2021). Furthermore, optimization algorithms such as Particle Swarm Optimization (PSO) have been used to enhance model performance by tuning hyperparameters and improving convergence behavior (Albadr et al., 2024; Maheshwari & Begde, 2025).

In addition to machine learning advancements, the integration of optimization techniques has shown significant potential in improving model efficiency and accuracy. Metaheuristic algorithms, including PSO and other evolutionary approaches, have been widely applied to optimize feature selection, model parameters, and classification performance (Albadr et al., 2024). These algorithms are particularly effective in navigating complex search spaces and identifying optimal solutions in high-dimensional problems. Similarly, Extreme Learning Machines (ELMs) have emerged as a fast and efficient alternative to traditional neural networks, offering rapid training and good generalization performance (Yu et al., 2015). However, ELM models may suffer from instability due to random initialization of parameters, which necessitates the use of optimization techniques to enhance their reliability.

Despite the significant progress in this field, several challenges remain unresolved. First, the scarcity of publicly available and well-labeled datasets limits the ability to train and evaluate machine learning models effectively. Second, the presence of noise, sparsity, and heterogeneity in feedback data complicates the detection process. Third, existing methods often fail to achieve a balance between detection accuracy, computational efficiency, and scalability. Moreover, many studies focus on either textual analysis or behavioral patterns, without integrating both aspects into a unified framework (Ennaouri & Ahmed, 2022; Miradi et al., 2025). These limitations highlight the need for more comprehensive and adaptive approaches that can effectively address the complexities of fake feedback detection in cloud environments.

Furthermore, the increasing sophistication of fraudulent behaviors necessitates the development of models capable of detecting both individual and group-based anomalies. Recent studies have emphasized the importance of analyzing user interactions, temporal patterns, and network structures to identify coordinated attacks and collusive behaviors (Goswami et al., 2017; Liu et al., 2019). In addition, integrating multiple data sources, such as textual content, numerical ratings, and user metadata, can provide a more holistic view of the problem and improve detection accuracy (Zhang et al., 2020; Zhang et al., 2022). However, achieving such integration while maintaining computational efficiency remains a significant challenge.

Given these considerations, there is a clear need for a robust, scalable, and accurate framework that can effectively detect fake feedback in cloud environments by leveraging the strengths of clustering, statistical analysis, and machine learning. The integration of hierarchical clustering techniques to address data dispersion, advanced statistical methods for anomaly detection, and optimized machine learning models for classification offers a promising solution to this problem. Such a hybrid approach can enhance the detection of complex patterns, improve model stability, and reduce classification errors, thereby contributing to the development of more reliable and trustworthy cloud systems (Liu & Pang, 2018; Martens & Maalej, 2019).

Therefore, the aim of this study is to propose a hybrid framework based on improved agglomerative hierarchical clustering, enhanced empirical data analytics, and an Extreme Learning Machine optimized by Particle Swarm Optimization to accurately detect fake feedback and improve trust in cloud computing environments.

2. Methods and Materials

To address the problem of fake feedback detection in cloud environments, the proposed method is applied to identify any type of anomaly, including outliers and fraudulent data, within the target dataset. In this study, the focus is on user feedback regarding services obtained from cloud service providers. For these services, users may utilize one or multiple providers to execute their applications.

Accordingly, the main challenges addressed in this research include: detecting fake feedback in cloud environments; the limited number of studies in this domain; the lack of publicly available datasets with appropriate characteristics; the absence of labeled classes for fake feedback; and the high dispersion of feedback data. Therefore, the proposed method for fake feedback detection is based on data labeling and statistical analysis, as illustrated in Figure 1. In the figure, innovative components and the application of suitable algorithms are indicated with dashed lines. The stages of the proposed method are as follows:

First stage: utilization of a standard dataset.

Second stage: preprocessing and sorting of the dataset file.

Third stage: application of improved agglomerative hierarchical clustering combined with the Reptile Search Algorithm (RSA) to address data dispersion and cold-start problems.

Fourth stage: dataset analysis using an enhanced empirical data analysis method that integrates dual-scale density estimation, multi-faceted analysis with frequency weighting, and Chebyshev-based anomaly filtering to improve anomaly detection performance.

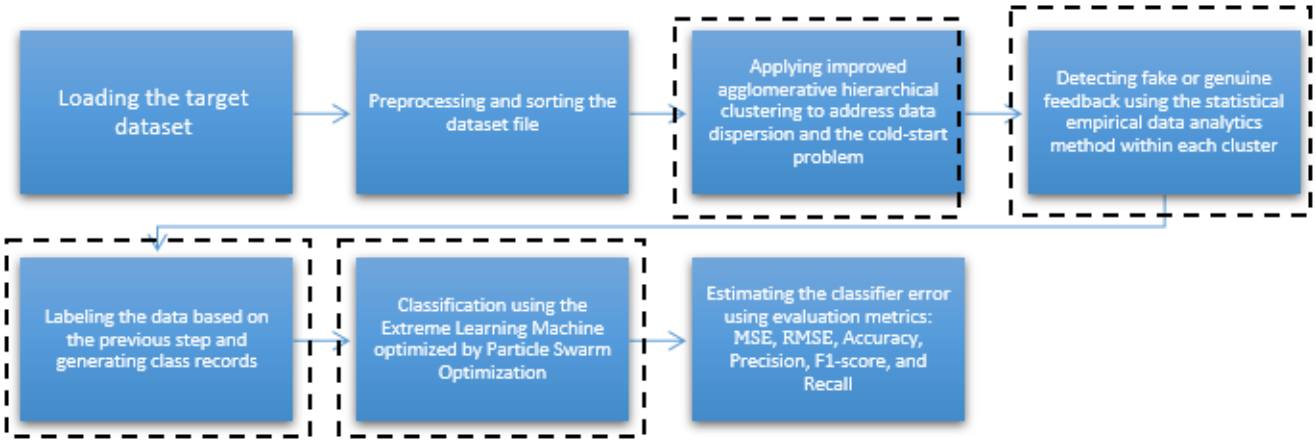
Fifth stage: detection of fake feedback within the dataset and generation of class records.

Sixth stage: classification using an Extreme Learning Machine optimized with Particle Swarm Optimization.

Seventh stage: estimation of classifier error using standard evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Accuracy, Recall, Precision, and F1-score.

Figure 1

Workflow of the proposed method



The algorithms and procedures required for implementing the proposed method are described as follows:

2.1. Preprocessing

To apply the proposed method, preprocessing must be performed on the dataset. The preprocessing steps considered in this study are illustrated in Figure 2.

Sorting and feature separation: For analytical purposes, the relevant features for each user are extracted, organized, and structured into specific columns. For example, in the dataset, features are stored in the format (51908, 182, 4, 'Helpful', 12223, '2001-02-09') for each user.

Removal of irrelevant columns: Temporal columns that are not essential for the analysis are removed.

Discretization: Textual values are converted into numerical representations, such that "Not Yet Rated" = 1, "Somewhat Helpful" = 2, "Helpful" = 3, "Very Helpful" = 4, and "Show" = 5.

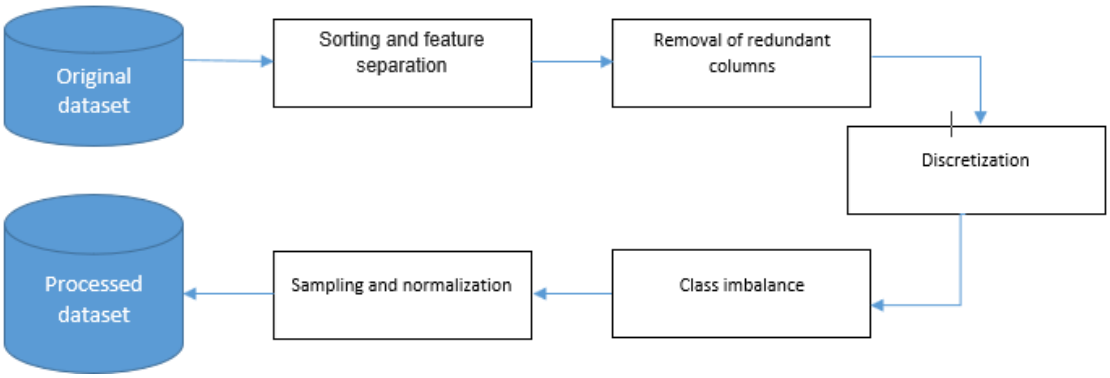
Class imbalance: One of the main challenges in this study is the limited number of anomalous instances, resulting in an imbalanced data distribution where anomalous data are significantly fewer than normal data. This imbalance may bias machine learning models toward the majority class and reduce predictive accuracy. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) is employed. By generating synthetic samples for the minority class (anomalous data), the data distribution is balanced.

Sampling and normalization: For the application and evaluation of the Particle Swarm Optimization-based

Extreme Learning Machine, the dataset is randomly divided into training (70%) and testing (30%) subsets. Additionally, prior to being fed into the learning algorithm, the data are normalized within the range $[0, 1]$.

Figure 2

Preprocessing steps applied to the dataset



2.2. Improved Agglomerative Hierarchical Clustering Using the Reptile Search Algorithm

By clustering users with similar opinions, feedback dispersion can be reduced and fake feedback detection can be facilitated. In this study, an improved agglomerative hierarchical clustering approach is employed. Unlike K-Means clustering, in this method each observation may belong to multiple clusters at different hierarchical levels, as clusters are formed based on varying distance thresholds. Consequently, each cluster may be a subset of another cluster at a specific distance level. In this approach, each data point is initially considered as an individual cluster, and through an iterative process, clusters with higher similarity are merged until a single cluster is formed. The proposed algorithm is a hybrid of agglomerative hierarchical clustering and the Reptile Search Algorithm, described as follows:

Agglomerative hierarchical clustering: This is a widely used unsupervised clustering method that represents the hierarchical structure of data as a tree (dendrogram). Initially, each sample is treated as an independent cluster, and at each step, the two clusters with the highest similarity (or smallest distance) are merged. This process continues until all samples are grouped into a single cluster. The choice of distance metric and feature weighting plays a critical role in clustering quality. In the classical AHC method, similarity or distance between samples is calculated using metrics such as Euclidean distance:

$$\text{dist}(x_i, x_j) = \sqrt{\sum_{u=1}^{n} | x_{iu} - x_{ju} |^2}$$

Afterward, samples are recursively merged to form a hierarchical cluster structure. The main limitation of this method is that it assigns equal weights to all features and uses a fixed, predefined distance metric. However, in real-world datasets, some features are more important than others, and the choice of distance metric can significantly affect clustering performance. To overcome this limitation, the Reptile Search Algorithm is employed to optimally determine feature weights and adjust the distance metric.

Reptile Search Algorithm: The Reptile Search Algorithm is a recent metaheuristic method introduced by Abualigah et al. in 2022, inspired by the hunting behavior and movement patterns of reptiles in nature. This algorithm is designed based on two key behavioral phases: exploration and exploitation. In the exploration phase, the reptile moves across different regions of the search space to discover new areas, while in the exploitation phase, it focuses on promising regions and performs a final attack to identify the optimal solution.

The process of the algorithm is described as follows. Let N denote the population size, T the total number of iterations, and X_i^t the position of the i-th reptile at iteration t. The search space is bounded by lower and upper limits L_b and U_b, respectively. During the initialization phase, the position of each reptile is randomly generated within the search space according to Equation (1):

$$(1) X_i^0 = L_b + \text{rand}(0,1) \times (U_b - L_b)$$

After initialization, each position is evaluated using the objective function $f(X_i)$to determine the best initial position X_{best}^0. The Reptile Search Algorithm (RSA) alternates between two main phases at each iteration. In the exploration phase, the movement of a reptile to explore new regions is modeled as follows in Equation (2):

$$(2)X_i^{t+1} = X_r^t + \beta \cdot (X_i^t - X_r^t) + \alpha_1 \cdot (X_{\text{best}}^t - X_r^t) \cdot \text{rand}(0,1)$$

where $X_r^t$is a randomly selected reptile from the population, $X_{\text{best}}^t$is the current best position, $\beta \in [0.5,1]$is a random coefficient controlling the movement intensity, and $\alpha_1 = 2(1 - t/T)$is a linearly decreasing coefficient that reduces exploration as iterations increase. The term $\text{rand}(0,1)$represents a uniformly distributed random value between 0 and 1, used to preserve population diversity. In the exploitation phase, the algorithm focuses on refining positions near the current best solution, as defined in Equation (3):

$$(3)X_i^{t+1} = X_{\text{best}}^t + F \cdot \beta \cdot (X_i^t - X_{\text{best}}^t) + \alpha_l \cdot \text{rand}(0,1) \cdot (U_b - L_b)$$

where $F \in [-1,1]$is a random factor that determines the direction and magnitude of the reptile's movement. The third term enables the algorithm to maintain a degree of randomness and avoid being trapped in local optima. After each update, positions that fall outside the allowable bounds are controlled using Equation (4):

$$(4)X_i^{t+1} = \min(\max(X_i^{t+1}, L_b), U_b)$$

At each iteration, the updated positions are evaluated using the objective function $F(X_i^{t+1})$, and the best position is updated accordingly. This process continues until the stopping criterion, defined as the maximum number of iterations T, is reached. In this study, the fitness function is defined based on the clustering validity index as follows:

$$(5)f(X_i^{t+1}) = \text{mean}(\text{silhouette}(X_i^{t+1}))$$

where $f(X_i^{t+1})$is the objective function to be maximized, and $\text{silhouette}(\cdot)$represents the silhouette index computed over the feature-weight vector optimized by the RSA algorithm. The silhouette index is employed as the fitness measure due to its stability in datasets with asymmetric distributions. In the proposed method, RSA is used to learn optimal feature weights and select an appropriate distance metric for hierarchical clustering. Each reptile represents a weight vector w, and the weighted distance matrix between samples is computed as follows:

$$(6)D_{i,j}(w) = \sqrt{\sum_{k=1}^{d} w_k \mid x_{ik} - x_{jk} \mid^2}$$

where x_{ik}denotes the value of the k-th feature for sample i, x_{jk}denotes the value of the k-th feature for sample j, and $x_{ik} - x_{jk}$represents the squared difference of the k-th feature between the two samples. By adaptively balancing exploration and exploitation phases and utilizing dynamic random coefficients, RSA achieves an effective trade-off between global search and local refinement. This capability enables RSA to deliver more stable and faster performance compared to classical algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) in clustering, feature selection, and multidimensional optimization tasks. The pseudocode of the improved agglomerative hierarchical clustering using RSA is presented in Table 1.

Table 1

Pseudocode of Improved Agglomerative Hierarchical Clustering (AHC + RSA)

Algorithm 1: AHC + RSA
Input: Data matrix $X$with dimensions $(N \times d) \leftarrow N$samples, $d$features
Step 1: Initialization
Determine the number of clusters K
Apply K-Means to reduce the dataset into centroids $C = \{C_1, C_2, \ldots, C_m\}$, where $N > m$
Set RSA parameters: Population size $PopSize$, Maximum iterations $MaxIter$
Initialize individuals $X_i = [w_1, w_2, \ldots, w_d]$randomly in $[0, 1]$
Normalize weights: $w_i = \frac{w_i}{\sum_{k=1}^{d} w_k}$
Step 2: Run RSA
For each iteration $t = 1$to $MaxIter$:
Compute $\alpha_1 = 2(1 - t/MaxIter)$

Set $\beta = 0.5 + \frac{\text{rand}()}{2}$
For each reptile $i = 1, \ldots, PopSize$:
If rand < 0.5(Exploration phase):
Select random X_r
Update: $X_i^{t+1} = X_r^t + \beta(X_i^t - X_r^t) + \alpha_1(X_{\text{best}}^t - X_r^t)\text{rand}(0,1)$
Else (Exploitation phase):
Set $F = 2\text{rand}(0,1) - 1$
Update: $X_i^{t+1} = X_{\text{best}}^t + F\beta(X_i^t - X_{\text{best}}^t) + \alpha_l \text{rand}(0,1)(U_b - L_b)$
Apply boundary control: $X_i^{t+1} = \min(\max(X_i^{t+1}, L_b), U_b)$
Normalize weights
Compute weighted distance: (D_{i,j}(w) = \sqrt{\sum_{k=1}^{d} w_k
Apply AHC using Ward linkage
Partition into Kclusters and compute fitness: $f(X) = \text{mean}(\text{silhouette}(X,\text{labels}))$
End For
Select best solution $f(X_{\text{best}})$
End For
Step 3: Final Clustering
Compute optimal weights w_{best}
Apply AHC on centroid matrix Cusing weighted features
Generate final cluster labels $L = \{L_1, L_2, \ldots, L_N\}$
Compute clustering quality metrics (e.g., Davies–Bouldin index, silhouette score)
Output: Final labels L, optimal weights w_{best}, final fitness f_{best}

In the time complexity analysis of the hybrid algorithm combining agglomerative hierarchical clustering and RSA, two main components are considered. Since RSA is a population-based algorithm, a complete hierarchical clustering process must be executed for each population member in every iteration. Therefore, the overall time complexity is the product of the number of iterations, population size, and the computational cost of AHC. For a dataset with $m$samples and $d$features, the computation of the distance matrix in AHC requires $O(m^2 d)$, and the hierarchical merging process requires $O(m^2 \log m)$in the best case. Additionally, the silhouette index used as the objective function requires $O(m^2)$time for each evaluation. Consequently, the total time complexity of the hybrid algorithm can be expressed as:

$$(7) O((m^2 \log m + m^2 d) \cdot N \cdot MaxIter)$$

where $N$is the population size, $MaxIter$is the number of iterations, $m$is the number of samples, and $d$is the number of features. As the dataset size increases, particularly when $m$becomes large, the dominant computational cost arises from the quadratic operations in AHC and silhouette evaluation. Therefore, for large-scale datasets, approximate methods or dimensionality reduction and sampling techniques are typically employed to reduce computational overhead and improve practical feasibility. In the worst-case scenario for hierarchical clustering, all users may be grouped into a single cluster, meaning that users who provided similar feedback to cloud service providers are clustered together, or conversely, each user may form an individual cluster, resulting in as many clusters as users. In such cases, based on Step 4 of the proposed method, the enhanced empirical data analysis approach, which is a robust statistical method, is applied to the entire dataset.

2.3. Enhanced Empirical Data Analytics (E-EDA) Method

In this study, Euclidean distance and density distribution are used to identify anomalies and improve clustering performance. The input dataset is defined as $x_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\} \in \mathbb{R}^n$, where x_{i1}denotes the i-th data sample, $k$is the total number of samples, $\{u_1, u_2, \ldots, u_l\} \in \mathbb{R}^n$is the set of unique samples, and $\{f_1, f_2, \ldots, f_l\}$represents the frequency of occurrence of each u_i, such that $\sum_{i=1}^{l} f_i = k$and $l \leq k$. The output consists of the set of potential anomalies within each cluster. The proposed method includes four main stages.

Stage 1: Computation of Cumulative Proximity
For each data point x_i, cumulative proximity is defined as the sum of squared distances from all other data points:

$$\pi(x_i) = \sum_{j=1}^{k} \| x_i - x_j \|^2 = K(\| x_i - \mu \|^2 + X - \| \mu \|^2) \quad (8)$$

where $\| x_i - x_j \|^2$is the Euclidean distance between two data points. Using the sample mean $\mu = \frac{1}{K}\sum_{l=1}^{n} x_i$and the mean squared norm $X = \frac{1}{K}\sum_{l=1}^{n} \| x_i \|^2$, this measure represents the overall dispersion of the data around the mean.

Innovation: In this study, $\pi(x_i)$is used not only for measuring global distance but also as a basis for defining local and multimodal densities.

Stage 2: Computation of Unimodal Density
The unimodal density for each data point is defined as follows:

$$D^{UM}(x_i) = \frac{\sum_{k=1}^{K} \pi(x_k)}{2K\pi(x_i)} = \frac{\sum_{k=1}^{K} \sum_{j=1}^{K} \| x_k - x_i \|^2}{2K \sum_{j=1}^{K} \| x_i - x_j \|^2} = \frac{1}{1 + \frac{\| x_i - \mu \|^2}{X - \| \mu \|^2}} \quad (9)$$

The factor of 2 in the denominator of $D^{UM}$is used because each pairwise distance is counted twice in the total density sum over all data samples. This expression has the form of a Cauchy function and indicates the concentration of data around the mean. If the distribution is symmetric, the tails of the distribution are balanced. If the right tail is more extended, the distribution is right-skewed. If the left tail is more extended, the distribution is left-skewed.

Innovation: Unlike the original EDA version, in this study unimodal density is used to extract a data asymmetry index and improve outlier detection.

Stage 3: Computation of Multimodal Density
For each unique data point $u_i$with occurrence frequency f_i:

$$D^{MM}(u_i) = f_i \cdot D^{UM}(u_i) = \frac{f_i}{1 + \frac{\| u_i - \mu \|^2}{X - \| \mu \|^2}} \quad (10)$$

and for repeated data points where $x_j = u_i$:

$$D^{MM}(u_i) = f_i \cdot D^{UM}(x_j) \quad (11)$$

This stage enables the detection of local structures and hidden clusters without requiring iterative search algorithms.

Innovation: The use of frequency weighting in multimodal density calculation allows low-frequency outliers to be distinguished more rapidly than high-frequency data points.

Stage 4: Use of Chebyshev's Inequality and Local Density
For non-normal data, Chebyshev's inequality is applied:

$$p(x - E(x) \geq nx) \leq \frac{1}{n^2} \quad (13)$$

Accordingly:

1. The mean squared Euclidean distance is computed as follows:

$$d^2 = \frac{\sum_{k=1}^{K} \pi(x_k)}{K^2} = 2(X - \| \mu \|^2) \quad (14)$$

For each data point u_i, a hypersphere with radius $\bar{d}/2$is defined as the local influence region.

2. The local neighbors $\{u_i^l\}$are identified, and the weighted local density is calculated as:

$$D^{wL}(u_i) = \frac{(N_i - 1)}{L} \cdot f_i \cdot D^{L}(u_i) \quad (15)$$

where $N_i$is the number of unique neighbors and $L$is the total number of unique samples.

4. The data points are sorted according to $D^{wL}(u_i)$, and the portion with the lowest values, specifically those below $1/n^2$, is identified as the set of potential anomalies $\{x\}_2^{PA}$.

The combined use of weighted local density and Chebyshev's inequality reduces detection error and improves the accuracy of outlier extraction. The pseudocode of the empirical data analytics method is presented below:

Table 2

Pseudocode of the Empirical Data Analytics Method

Algorithm 2: Enhanced Empirical Data Analytics (E-EDA)
Input: Clustered samples
1. Start
2. For each cluster C_i, $i = 1, \ldots, K$:
a. Compute the local and global multimodal density for each sample: $D^{MM}(x_j) = f_{\text{local}}(x_j) + f_{\text{global}}(x_j)$
where $f_{\text{global}}$is approximated by Mahalanobis distance and $f_{\text{local}}$by LOF.
b. Identify anomaly candidates: samples with the lowest $D^{MM}(x_j)$.
c. Filter candidates according to local maxima to remove noise.
d. Update clusters by removing or marking anomalies.
3. End
Output: Improved clustering and detected anomalies

2.4. Extreme Learning Machine Optimized by Particle Swarm Optimization

In this study, an Extreme Learning Machine (ELM) optimized using the Particle Swarm Optimization (PSO) algorithm, as illustrated in Figure 4, is employed. The Extreme Learning Machine refers to a technique in which only $n$linear operations are required to update the weights, where $n$denotes the number of input data points. ELM is a type of neural network with only one hidden layer, and its difference from a conventional neural network lies in the fact that the first-layer weights remain unchanged during learning, while only the output-layer weights are trainable. In this algorithm, it is assumed that $n$arbitrary samples are available in the training phase in the form $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^m$, $i = 1,2, \dots, n$, with Lhidden nodes. The output function of the Extreme Learning Machine for a single-hidden-layer feedforward neural network with $n$hidden nodes is expressed as $f_n(x_j) = \sum_{i=1}^{L} \beta_i \, h_i(x_j) = \sum_{i=1}^{L} \beta_i \, G(a_i, b_i, x)$, $j = 1,2, \dots, N$. In this expression, $a_i$is the input weight vector connected to the i-th hidden-layer node, $b_i$is the bias value of the hidden-layer nodes, $\beta_i$is the output weight vector connected to the hidden-layer node, and $h_i(x) = \{h_1(x), h_2(x), \dots, h_i(x)\}$denotes the nonlinear feature mapping of the Extreme Learning Machine. The node output functions do not necessarily have to be unique and may be used in different hidden neural cells. Furthermore, $h_i(x) = G(a_i, b_i, x)$, where $a_i \in \mathbb{R}^d$and $b_i \in \mathbb{R}$, refers to the output functions of the hidden nodes with hidden-node parameters (a, b), and is a continuous nonlinear function satisfying the universal approximation property of the Extreme Learning Machine. In this study, the sigmoid activation function is adopted because of its extensive use in prior studies and its broad recognition among activation functions. The sigmoid function, also referred to as the logistic function, has an S-shaped form that can map any value to the interval $[0, 1]$. If the curve approaches positive infinity, the prediction is mapped to 1, whereas if the curve approaches negative infinity, the prediction is mapped to 0. In the Particle Swarm Optimization algorithm, each particle is regarded as a bird, and all birds seek to locate the best point through information sharing. This information-sharing process operates such that all birds compute their fitness function, and whichever bird attains the best fitness informs the others, thereby causing the remaining birds to move toward it. This process continues until all birds converge at the optimal point. Each bird selects the best movement direction according to its current position, its own previous best local position, and the best position of the other birds. In fact, the hypothesis of information sharing among birds constitutes the fundamental basis of this algorithm. By using PSO, the values of the input weights and the hidden-layer biases in the ELM algorithm are optimized. Let there be Narbitrary sample groups of (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbb{R}^n$and $y_i = [y_{i1}, y_{i2}, \dots, y_{im}]^T \in \mathbb{R}^m$, such that $x_i$is the input and $y_i$is the expected output.

Step 1: Data preprocessing. All features of the classification dataset are normalized to the range $[0, 1]$, after which the classification dataset is randomly divided into training and testing subsets according to the specified ratio.

Step 2: Pparticles are randomly initialized within the range $[-1,1]$for the hidden-neuron bias values and $[-1,1]$for the input-weight values, where $p = \{p_1, p_2, \dots, p_z\}$and $z$is the population size. The position of the i-th particle is defined as $p_i = \{w_{11}, w_{12}, \dots, w_{1n}, w_{21}, w_{22}, \dots, w_{2n}, \dots, w_{l1}, w_{l2}, \dots, w_{ln}, b_1, \dots, b_l\}$. The velocity of the i-th particle is defined as $v_i = \{v_{i1}, v_{i2}, \dots, v_{id}\}$, where $d = (1 + n) \times l$, and the velocity of each particle is bounded within $[v_{\min}, v_{\max}]$. The maximum number of iterations, denoted by $k_{\max}$, is given in Table 3.

Table 3

Variables Required for the Maximum Number of Iterations k_max

Algorithm Variables	Description
$w_{ij} \in [-1,1]$	Input-weight values representing the connection between the j-th input neuron and the i-th hidden neuron
$b_i \in [0,1]$	Bias of the i-th hidden neuron
n	Number of input neurons
l	Number of hidden neurons
$(1 + n) \times l$	Dimension of the particle requiring parameter optimization
W	Inertia weight
$c_1$and c_2	Acceleration coefficients
$r_1$and r_2	Random numbers in the range $[0, 1]$

Step 3: At this stage, the number of hidden neurons and the activation function of the Extreme Learning Machine are determined such that the objective function can be expressed as Equation (16):

$$f(p) = \sqrt{\frac{\sum_{i=1}^{N} \| \sum_{k=1}^{L} \beta_k \ f(w_k x_j + b_k) - t_j \|_2^2}{N}} \quad (16)$$

where $N$is the number of samples, $t_j$is the target output for the corresponding sample, $a_i$is the input weight vector connected to the hidden-layer node, $b_i$is the bias value of the hidden-layer nodes, and $\beta_k$is the output weight vector connected to the hidden-layer node. According to Equation (17), $\beta$is computed as follows:

$$\beta = H^{\dagger}T \quad (17)$$

where $H$and $T$are calculated according to Equation (18):

$$H = \begin{bmatrix} f(w_1 x_1 + b_1) & \cdots & f(w_l x_1 + b_l) \\ \vdots & \ddots & \vdots \\ f(w_1 x_N + b_1) & \cdots & f(w_l x_N + b_l) \end{bmatrix}_{N \times l}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_l^T \end{bmatrix}_{l \times m}, T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m} \quad (18)$$

where $H$is the hidden-layer output matrix of the Extreme Learning Machine network. In H, column $i$corresponds to the i-th hidden-layer neuron relative to the input neurons. $H^{\dagger}$denotes the Moore–Penrose generalized inverse of H. The activation function $f$is infinitely differentiable when the required number of hidden neurons satisfies $L \leq N$.

Step 4: At this stage, according to Equation (12), the value of each particle in population $p$is computed.

Step 5: At this stage, the best position of each particle, P_{localid}, as well as the best global position for all particles, p_f, is computed. The objective function is then used to update particles according to Equation (19):

$$P_{\text{localid}} = \begin{cases} p_i, & \text{if } p_i \text{ is better than } P_{\text{localid}} \\ P_{\text{localid}}, & \text{otherwise} \end{cases}$$

$$p_f = \begin{cases} p_i, & \text{if } p_i \text{ is better than } p_f \\ p_f, & \text{otherwise} \end{cases} \quad (19)$$

Step 6: The position and velocity of each particle are recalculated according to Equations (20) and (21), based on Equation (16):

$$v_{id}^{k+1} = W v_{id}^{k} + c_1 r_1 \left(P_{\text{localid}}^{k} - p_{id}^{k}\right) + c_2 r_2 \left(p_{fd}^{k} - p_{id}^{k}\right) \quad (20)$$

$$p_{id}^{k+1} = p_{id}^{k} + v_{id}^{k+1}, \forall i = 1,2,\ldots,z, \forall d = 1,2,\ldots,D \quad (21)$$

The constants $c_1$and $c_2$in Equation (20) are the learning parameters representing the degree of influence of the best particle positions. $k$is the loop counter, $W$is the parameter controlling the inertia of particle motion, and $r_1$and $r_2$are random numbers in the interval $[0, 1]$.

Step 7: If the stopping criteria are satisfied, the optimal input weights and hidden-layer biases are stored. This process is performed through optimization based on the minimum error value of the algorithm; otherwise, the procedure returns to Step 4.

Step 8: The results obtained from Particle Swarm Optimization are used as the input weights and biases for the Extreme Learning Machine, and the hidden-layer output matrix $H$is calculated using Equation (15).

Step 9: The output weight $\beta$is calculated based on Equation (13), and the predictive model of the Extreme Learning Machine is obtained with the minimum MSE evaluation criterion.

The pseudocode of the Extreme Learning Machine optimized by the Particle Swarm Optimization algorithm is presented in Table 4.

Table 4

Pseudocode of the Extreme Learning Machine Optimized by Particle Swarm Optimization

Algorithm 3: Extreme Learning Machine Optimized with Particle Swarm Optimization
Input: Dataset
1. Start
2. For $i = 1$to max_iterations: dataset partitioning and feature normalization in the range $[-1,1]$
3. Generate initial particles and initialize the position and velocity of each particle
4. Determine the number of hidden neurons and define the activation function of the Extreme Learning Machine
5. Calculate the value of each particle in population p
6. Calculate the best position of each particle and the best position for all particles
7. Calculate the position and velocity of each particle
8. If optimal input weights and hidden-layer biases of the Extreme Learning Machine are obtained
9. End

10. Calculate the hidden-layer output matrix H
11. Calculate the output weight β
a. If fitness(i) < elm_fitness(global_best, X_{train}, Y_{train},hidden_neurons)
b. global_best = particles$(i,:)$
c. BestMSE = fitness(i)
d. End
12. End
Output: Computation of evaluation criteria

3. Findings and Results

3.1. Simulation Environment

In this article, the simulation environment was implemented using MATLAB R2020a for modeling purposes, and the computer used for the simulation was equipped with an Intel Core i3-2350 CPU, 4 GB of memory, and a 500 GB hard drive.

3.2. Datasets

In this study, two datasets were used. The first dataset is Rich Epinions, which was employed in the studies of Seyadat et al. (Siadat et al., 2017) and Soleymani et al. (Soleymani et al., 2021). It was released by Simon Meyffret, Frédérique Laforest, and Lionel Médini under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The dataset was obtained from the website https://projet.liris.cnrs.fr/red and has been available on that site since June 2011. This dataset contains several tables, including the user expertise table for categories, the product category table, the user table, the feedback table, the feedback evaluation table, the similarity table, and the trust table between users. The second dataset is CloudArmor, which was used in the studies of Truong et al. (Truong et al., 2023) and Aghaee Qazvini et al. (Aghaee Ghazvini et al., 2020), and is one of the limited datasets available with respect to cloud service providers and cloud customers. This dataset includes feedback from cloud customers on multiple quality-of-service parameters related to each cloud service provider, along with current-time or event-time information associated with each feedback entry. Due to the unavailability of the real CloudArmor dataset—which had been introduced in previous years by the original authors and is currently inaccessible—a simulated synthetic dataset was used in this study to enable the evaluation of the proposed method. The structure of these data was designed based on the published characteristics and descriptions of CloudArmor. Specifically, the synthetic dataset includes user ID, service provider ID, numerical rating assigned to the service, feedback submission time, and feedback text representing the user's qualitative opinion regarding service quality, such as "Excellent service" or "Poor reliability." These textual feedback records were converted into corresponding numerical values to make them usable in machine learning models. In addition, to simulate more realistic conditions, a percentage of the feedback entries was injected as fake or unrealistic so that the proposed method could also be evaluated in terms of its capability to detect such data. This design makes it possible to examine the model's performance in an environment close to real cloud-service data. This dataset includes 12,000 feedback records from 5,000 users and 113 services. Out of the 12,000 records, 1,800 were considered fake feedback records. Fake users typically submit multiple similar feedback entries on different services, use short and exaggeratedly positive expressions, and assign extremely high or extremely low ratings (1 or 5). Real users exhibit more diverse patterns and use more natural expressions.

3.3. Evaluation Criteria

In this study, the evaluation criteria used in various previous studies (Cho et al., 2023; Deshai & Rao, 2023; Javed et al., 2021; Siadat et al., 2017; Soleymani et al., 2021; Taneja & Kaur, 2021; Thirunavukkarasu et al., 2023), were also employed, as presented in Table 5.

Table 5

Evaluation Criteria for the Performance of the Proposed Model

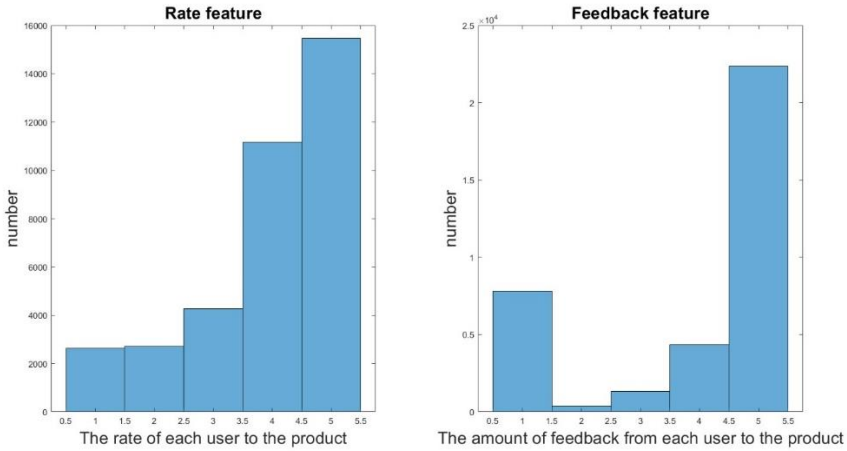
Criterion	Scientific–Practical Definition	Scale	Measurement Method
TP	Instances whose true class is positive and which are correctly classified as positive by the classification algorithm	Confusion Matrix	Numerical
TN	Instances whose true class is negative and which are correctly classified as negative by the classification algorithm	Confusion Matrix	Numerical
FP	Instances whose true class is negative but which are incorrectly classified as positive by the classification algorithm	Confusion Matrix	Numerical
FN	Instances whose true class is positive but which are incorrectly classified as negative by the classification algorithm	Confusion Matrix	Numerical
Accuracy	This criterion indicates the percentage of the total test records that the designed classifier has classified correctly	Formula-based calculation	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
Precision	The percentage of positive predictions made by the algorithm that are correct; the main focus of this criterion is the correctness of the algorithm's positive predictions	Formula-based calculation	$\text{Precision} = \frac{TP}{TP + FP}$
F1-score	A method for evaluating the accuracy of a learning model obtained from the estimates of Recall and Precision	Formula-based calculation	$\text{F1-score} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
Recall	The proportion of positive instances that are correctly identified by the classifier as positive samples	Formula-based calculation	$\text{Recall} = \frac{TP}{TP + FN}$
MSE	A method for estimating the error magnitude, representing the difference between estimated values and actual values	Formula-based calculation	$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(P_i - A_i)^2$
RMSE	The difference between the value predicted by the model or statistical estimator and the actual value	Formula-based calculation	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - A_i)^2}$

3.4. Implementation of the Proposed Method and Required Parameters

To implement the proposed method, after loading the first dataset, Rich Epinions, containing 36,220 records with 6 features—feedback code, user code, rating value, feedback, product code, and time—the preprocessing operations were performed on it. The histogram of the two features, namely rating and users' feedback to the service provider, is presented in Figure 3.

Figure 3

Histogram of the two features, rating and feedback



As shown in Figure 6, most users assigned a rating of 5, and the feedback category was predominantly *Very Helpful*. The next step involved performing improved agglomerative hierarchical clustering using the RSA algorithm. The required parameters are presented in Table 6.

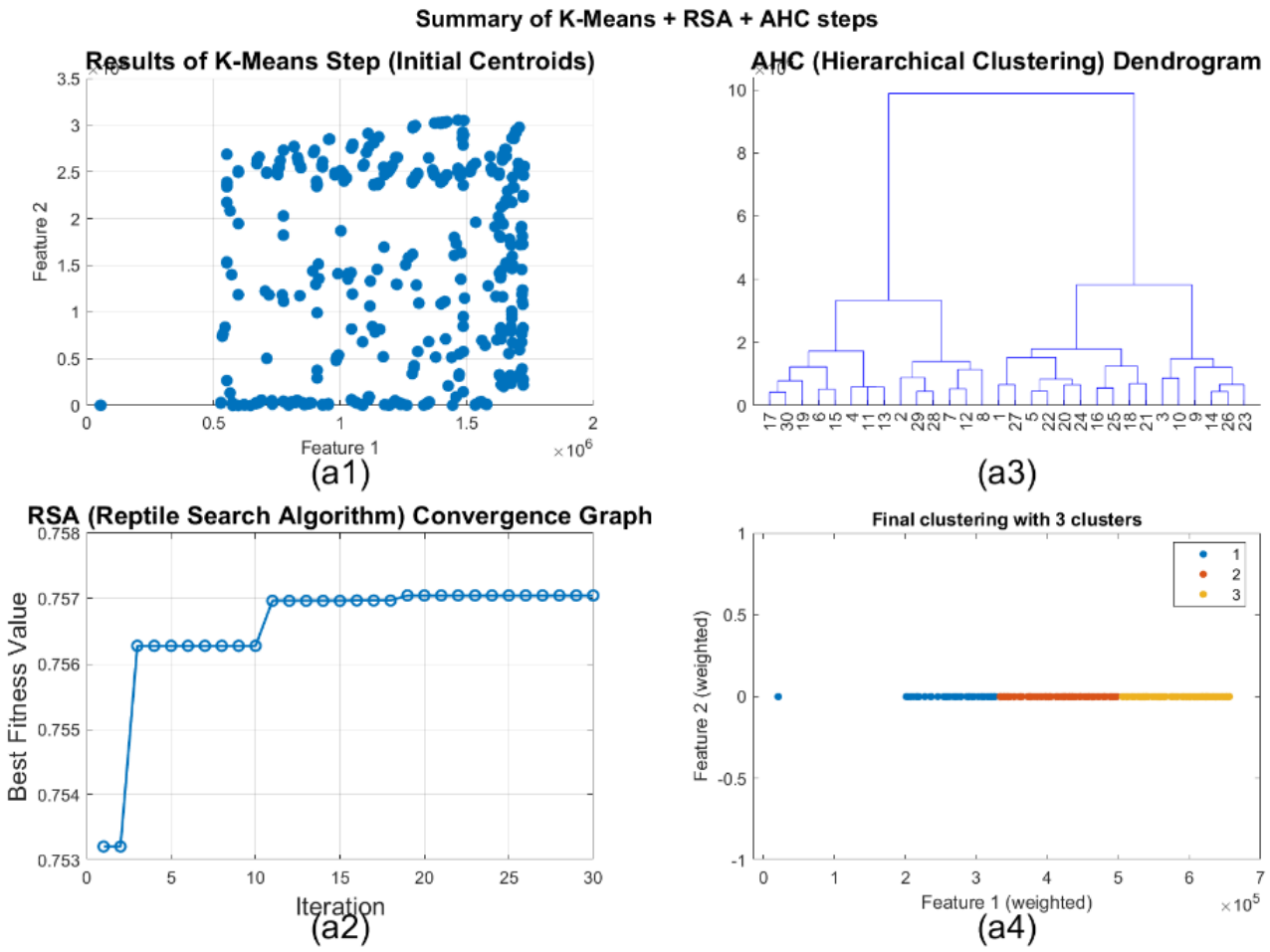
Table 6

Parameters of Improved Agglomerative Hierarchical Clustering Using the RSA Algorithm

Parameter Type	Parameter Name	Symbol / Variable	Value or Range	Description
Data input	Data matrix	X	$N \times d$	Includes Nsamples and $d$features
Preprocessing stage	Number of initial K-Means clusters	numPreClusters	500	Reduces the data volume from Nsamples to 500 initial centroids
RSA algorithm	Population size of flies	PopSize	20	Number of solutions in each iteration
	Maximum number of iterations	MaxIter	30	Number of search steps
	Problem dimension	dim	d(number of features)	Equal to the number of features in the cluster centroids
	Lower bound of weights	lb	0	Minimum value of each feature weight
	Upper bound of weights	ub	1	Maximum value of each feature weight
Objective function	Clustering evaluation index	Fitness	Mean Silhouette	Clustering quality based on intra-cluster homogeneity and inter-cluster separation
Final number of clusters	Number of clusters	numClusters	2 to 10 (random or optimal)	Number of clusters in each iteration of the algorithm

Figure 4

Clustering of the dataset using improved agglomerative hierarchical clustering with the RSA algorithm



As shown in Figure 4, in order to reduce data volume and avoid excessive memory consumption during the execution of the optimization algorithm, the K-Means algorithm was first used for data preprocessing. In this stage, the initial data

with Nsamples and $d$features were reduced to 500 cluster centroids so that the distributional structure of the data could be preserved while decreasing the input dimensionality for the optimization stage. These initial centroids were then used as representatives of the original data throughout the subsequent process. In the second step, the RSA algorithm was executed to determine the optimal feature weights and maximize the silhouette index, because in this study the objective function was defined as the maximization of clustering quality. The results obtained from the convergence plot of the algorithm showed that the fitness value increased rapidly and, after several iterations, reached a stable value close to 0.757, indicating effective convergence of the algorithm. In the final step, hierarchical clustering was performed on the weighted data using the Ward linkage method. The resulting dendrogram displayed the hierarchical structure of the clusters, and in the final outcome, the data were categorized into three distinct clusters with appropriate separability. The overall results indicate that the combination of K-Means for data reduction, the RSA algorithm for feature-weight optimization, and AHC for final clustering improved clustering stability and quality compared with the direct use of each method individually. In the final clustering stage, the weighted data, after optimization by the RSA algorithm, were represented in normalized form. For this reason, the feature values were concentrated around zero, and the cluster distribution appeared compact along the horizontal axis of the plot. This was caused by data standardization after applying the weighting coefficients and does not indicate any error in the process.

Furthermore, in the clustering process, because users with similar characteristics and viewpoints are placed in the same cluster, the cluster label can be used as a new feature in the dataset. This feature can reflect users' collective behavior and hidden decision-making patterns and can therefore play an effective role in improving the performance of supervised learning models. In other words, assigning each record to a specific cluster provides a type of high-level knowledge regarding users' response patterns, and combining this information with other numerical and textual features increases the accuracy of prediction models and fake feedback detection. Table 7 shows the number of samples in each cluster and represents the data distribution across the three main clusters.

Table 7

Number of Samples in Each Cluster

Number of Clusters	Cluster 1	Cluster 2	Cluster 3
Number of samples in each cluster	1,934	17,709	16,577

Based on Table 7, Cluster 1, with 1,934 samples, has the smallest size, whereas Clusters 2 and 3, with 17,709 and 16,577 samples, respectively, contain the majority of the data.

The next stage involved using the enhanced empirical data analytics method to detect fake feedback within each cluster. Table 8 presents the number of detected fake feedback instances.

Table 8

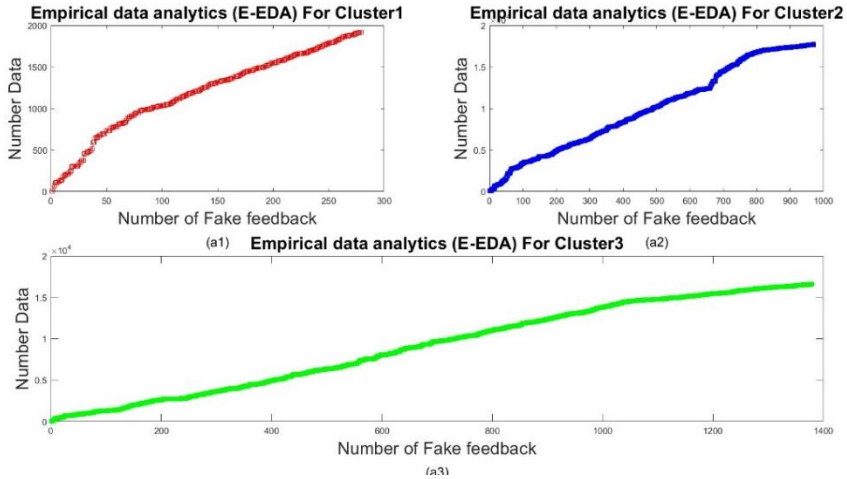
Number of Fake Feedback Records Detected by the Enhanced Empirical Data Analytics Method

Number of Clusters	Cluster 1	Cluster 2	Cluster 3
Number of fake feedback records in each cluster	279	970	1,379

According to Table 8, by using the enhanced empirical data analytics method, a total of 2,628 records were identified as fake feedback. Of these, 279 samples belonged to Cluster 1, 970 samples belonged to Cluster 2, and 1,379 samples belonged to Cluster 3, as illustrated in Figure 5.

Figure 5

Detection of fake feedback using the enhanced empirical data analytics method



As shown in Figure 5, the largest number of detected fake feedback records belongs to Cluster 3. The next stage is class-label assignment. In the dataset, records identified as anomalies and fake feedback are assigned to Class 1, while the remaining records are assigned to Class 0.

The next stage involved using the Extreme Learning Machine optimized by Particle Swarm Optimization for model training and evaluation. However, because the number of records in the normal class, or genuine feedback, is much larger than the number of records in the fake feedback class, the study faces the problem of class imbalance, which requires data balancing. In this study, the SimpleSMOTE method was used to address class imbalance. This method balances the data distribution by generating synthetic samples from the minority class and avoids the removal of real data. In the SimpleSMOTE process, for each sample in the minority class, a number of synthetic samples are generated by averaging and adding random noise in the feature space. The value of the oversampling parameter, which determines the number of synthetic samples, plays an important role in model performance; an excessively large value may lead to class overlap and reduced model accuracy, whereas a very small value may not have a substantial effect on improving data balance. In this study, this parameter was tuned empirically by testing several different values in order to obtain the best balance between precision and recall. After performing preprocessing and data balancing, the total number of samples decreased from 36,220 to 27,203, due to the removal of invalid data and class adjustment. In addition, to prevent data leakage and ensure a fair evaluation of model performance, the data were first divided into training (70%) and testing (30%) subsets, and then the SMOTE process was applied only to the training set. Table 9 presents the parameters considered for the Extreme Learning Machine optimized by Particle Swarm Optimization.

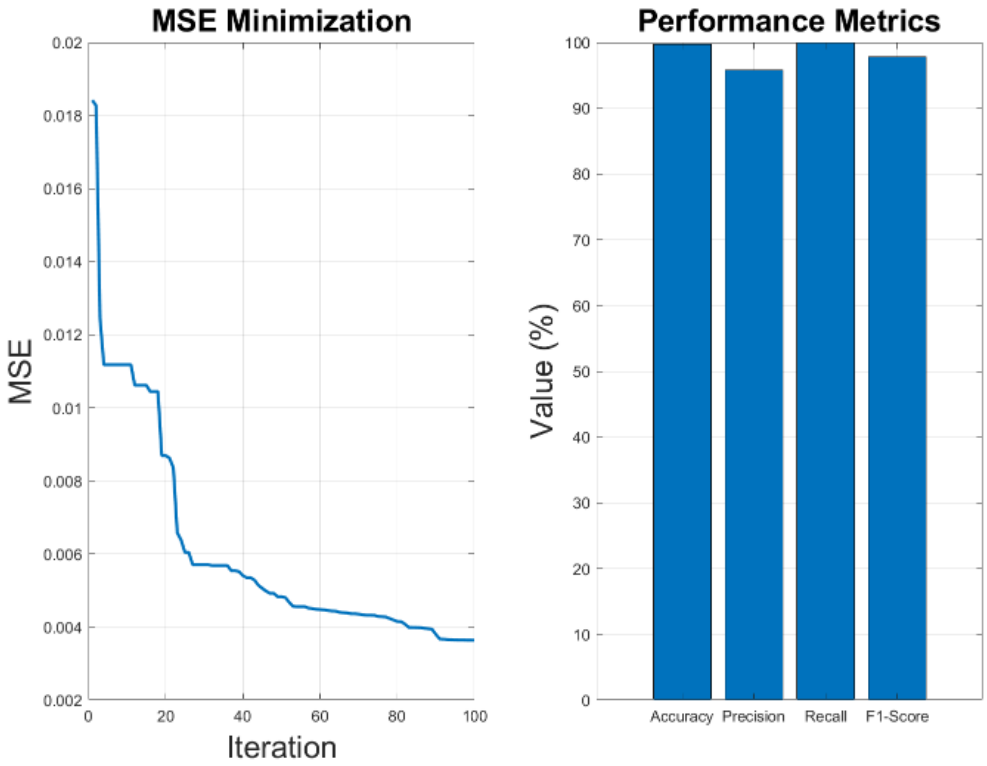
Table 9

Parameters of the Extreme Learning Machine Optimized by Particle Swarm Optimization

Input Value	Algorithm Variables
30	hidden_neurons
50	particle_count
100	max_iterations
1.99	$c_1 = c_2$
0.7	$r_1 = r_2$
size(dataset, 2)	input_size
hidden_neurons * (input_size + 1)	dimensions
reshape(particle(1:hidden_neurons * input_size), [hidden_neurons, input_size])	$w_{ij} \in [-1,1]$
particle(hidden_neurons * input_size + 1:end)	$b_i \in [0,1]$
tanh(X * input_weights' + biases')	H
pinv(H) * Y	B

Figure 6

Optimized MSE value at each iteration and evaluation results



As shown in Figure 6, the performance of the optimization algorithm is presented in two parts: the "error-value convergence process" and the "evaluation results." The left-hand plot shows the trend of Mean Squared Error (MSE) reduction over 100 iterations. The MSE value decreases from approximately 0.018at the beginning of the process to less than 0.003around the seventieth iteration, after which it stabilizes at an approximately constant value. This behavior indicates that the algorithm has successfully reached the minimum possible error and achieved stable convergence. The right-hand plot presents the other evaluation criteria, while the complete results are reported in Table 10. This table shows the output of the Extreme Learning Machine optimized by the Particle Swarm Optimization algorithm based on the evaluation criteria.

Table 10

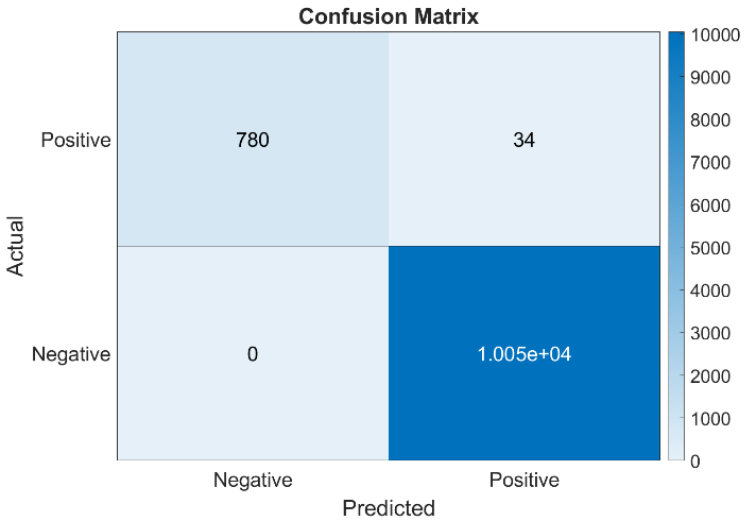
Output of the Extreme Learning Machine Optimized by Particle Swarm Optimization Based on the Evaluation Criteria

Evaluation Criteria	MSE	RMSE	Accuracy	Precision	Recall	F1-score
Epinions dataset	0.0036	0.0604	99.69%	95.82%	100%	97.87%

According to the evaluation criteria reported in Table 10, the proposed model demonstrated highly accurate and stable performance. An accuracy of 99.69%and a recall of 100%indicate that all positive samples were correctly identified. In addition, a precision of 95.82%and an average F1-score of 97.87%show that the model achieved an appropriate balance between sensitivity and precision. Moreover, the low values of RMSE = 0.0604and MSE = 0.0036indicate good convergence and very low classification error. Overall, these results demonstrate that the model was able to correctly identify and classify the patterns in the Epinions data with very high accuracy and minimal error. Figure 7 shows the confusion matrix.

Figure 7

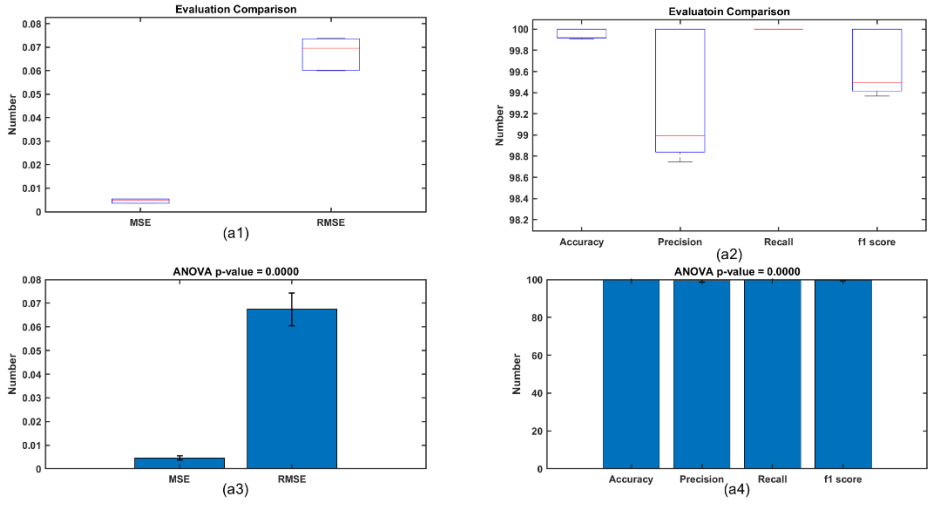
Confusion matrix



In Figure 7, together with Table 9, the confusion matrix of the proposed model is presented. According to the results, the proposed model performed very well on the Epinions dataset. The confusion matrix shows the results obtained from 30% of the test data, corresponding to approximately 11,000 samples out of a total of 27,168 samples, where the model correctly identified 10,050 true negative samples and 780 true positive samples, while only 34 positive cases were incorrectly predicted as negative, and no negative sample was incorrectly identified as positive. These results indicate the highly accurate performance of the model in distinguishing fake feedback from genuine feedback. The distribution of values in the matrix shows that the model achieved precision and recall values close to 100%and was able to learn the data pattern correctly with minimal classification error. This confirms the high effectiveness of the combination of the AHC-RSA_E-EDA_ELM-PSO algorithms in detecting fake feedback and improving prediction quality in cloud environments. In order to increase confidence in the validity of the results and prevent bias in performance analysis, ANOVA was conducted based on the evaluation criteria across 6 independent repetitions. Figure 8 shows the performance of the proposed method based on the ANOVA test.

Figure 8

Performance of the proposed method based on the ANOVA test



As shown in Figure 8, plots (a1) and (a3) correspond to the comparison of the MSE and RMSE errors and indicate that the values of both indices are very low, equal to 0.0044and 0.0662, respectively, with limited variance, which reflects the model's stability in prediction error. In addition, plots (a2) and (a4) show the comparison of the criteria of accuracy, precision, recall, and average F1-score. The mean accuracy over 10 iterations was 99.84%, precision was 98.84%, recall was 100%, and the F1-score was 99.36%, indicating that all these indices were above 98%. The *p*-value of 0.0000for both ANOVA tests also indicates the statistical significance of the results and a significant difference in the model's performance relative to the compared conditions. Therefore, the model has high stability and efficiency across multiple independent runs. Table 11 presents a comparison of the performance of the proposed method with the different methods used within it.

Table 11

Performance of the Proposed Method Based on Its Constituent Methods

Different Methods	MSE	RMSE	Accuracy	Precision	Recall	F1-score
Without clustering	No result, because this dataset requires 8.9 GB of memory	—	—	—	—	—
Proposed method with simple clustering (AHC-FFOA_E-EDA_ELM)	0.0037	0.0605	79.74%	68.88%	83.78%	75.60%
Proposed method with simple Extreme Learning Machine (AHC-FFOA_E-EDA_ELM)	0.4481	0.2109	96%	62.50%	83.33%	71.43%
Proposed method without E-EMD (AHC-FFOA_EDA_ELM-PSO)	0.0079	0.0886	96.91%	96.63%	100%	98.28%
Proposed method (AHC-RSA_E-EDA_ELM-PSO)	0.0036	0.0604	99.69%	95.82%	100%	97.87%

Based on the results obtained from Table 11, the use of hierarchical clustering reduced memory consumption and relatively improved model performance compared with the case without clustering. In the baseline case, execution of the model without clustering was not feasible because it required approximately 8.9 GB of memory. By applying the proposed method with simple agglomerative hierarchical clustering (AHC_E-EDA_ELM-PSO), the F1-score reached 75.60%and the overall accuracy reached 84.79%. By replacing it with the simple Extreme Learning Machine in the method (AHC-FFOA_E-EDA_ELM), the F1-score reached 71.43%and the overall accuracy reached 96%, which indicates improved convergence and greater model stability in learning the relationships among clusters. In addition, removing the E-EMD component also led to a relative decline in performance, whereas the final combination (AHC-FFOA_E-EDA_ELM-PSO), with an F1-score of 97.87%, recall of 100%, and the lowest MSE and RMSE values, achieved the best overall performance. This analysis shows that the addition of the optimization components (RSA) and advanced statistical analysis (E-EDA) directly improved the accuracy and stability of the model in clustering and fake feedback detection.

For further evaluation of the proposed method, it was also assessed on the second dataset, which corresponds to the cloud environment and was simulated based on the original CloudArmor sample. The only difference lies in the discretization of the feedback values, for which an encoding method was used in this dataset. Users' textual feedback was mapped to unique numerical values using the ordinal encoding method, such that each unique phrase in the variable Feedback_Text was replaced by a corresponding numerical identifier. This process was implemented in the MATLAB environment using the unique and strcmp functions, and its output was used as the variable Feedback_Code in the fake feedback detection model. Figure 9 shows AHC-RSA clustering on this dataset.

Figure 9

AHC-RSA clustering on the simulated CloudArmor dataset

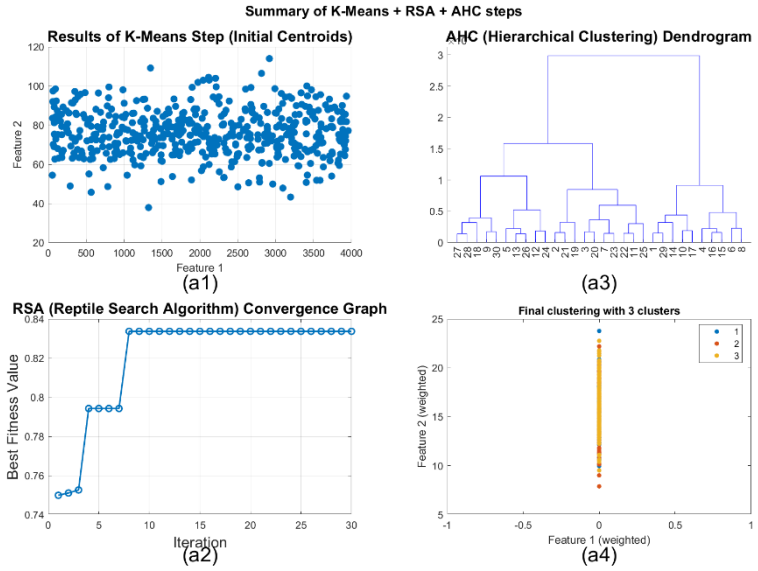
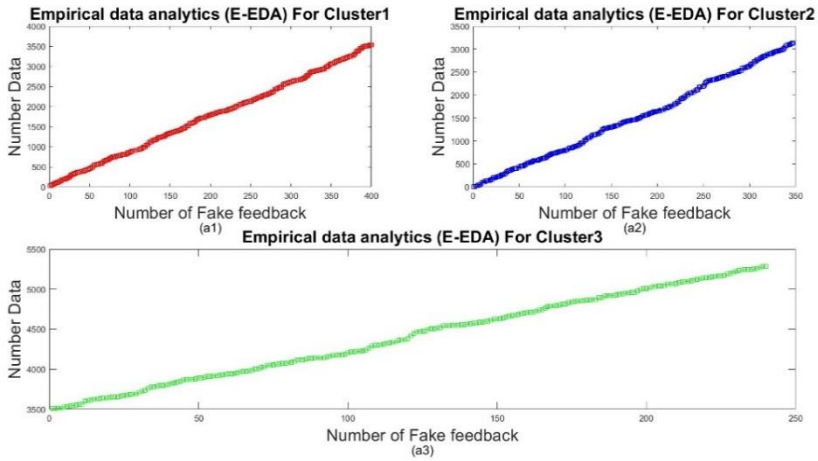


Figure 9 provides a summary of the stages of the proposed clustering process based on the combination of the K-Means, RSA, and AHC algorithms on the CloudArmor dataset. Plot (a1) shows the results of the initial K-Means stage, and plot (a2) shows the convergence trend of the RSA algorithm over 30 iterations, with stable convergence around 0.84. This trend indicates that the search process effectively converged to the optimal point and that the algorithm has high stability and efficiency. Plot (a3) presents the dendrogram of hierarchical clustering, and plot (a4) shows the final clustering result in the form of three distinct clusters. The distribution of the data in these three groups indicates high intra-cluster coherence and considerable separation between clusters, which demonstrates the high accuracy of the model in identifying the internal patterns of the data. Overall, the results shown in this figure indicate that the combination of the three algorithms, K-Means, RSA, and AHC, was able to identify and cluster the hidden structure of the CloudArmor cloud-environment data with rapid convergence and high stability. Figure 10 shows fake feedback detection with E-EDA on the simulated CloudArmor dataset.

Figure 10

Fake feedback detection on the simulated CloudArmor dataset



As shown in Figure 10, the enhanced empirical data analytics method identified a total of 987 records as fake feedback. Of these, 400 samples belonged to Cluster 1, 347 samples belonged to Cluster 2, and 240 samples belonged to

Cluster 3. This method was able to detect more than half of the fake feedback records. Figure 11 shows the trend of MSE reduction and the evaluation results for this dataset.

Figure 11

Trend of MSE reduction and evaluation results for the simulated CloudArmor dataset

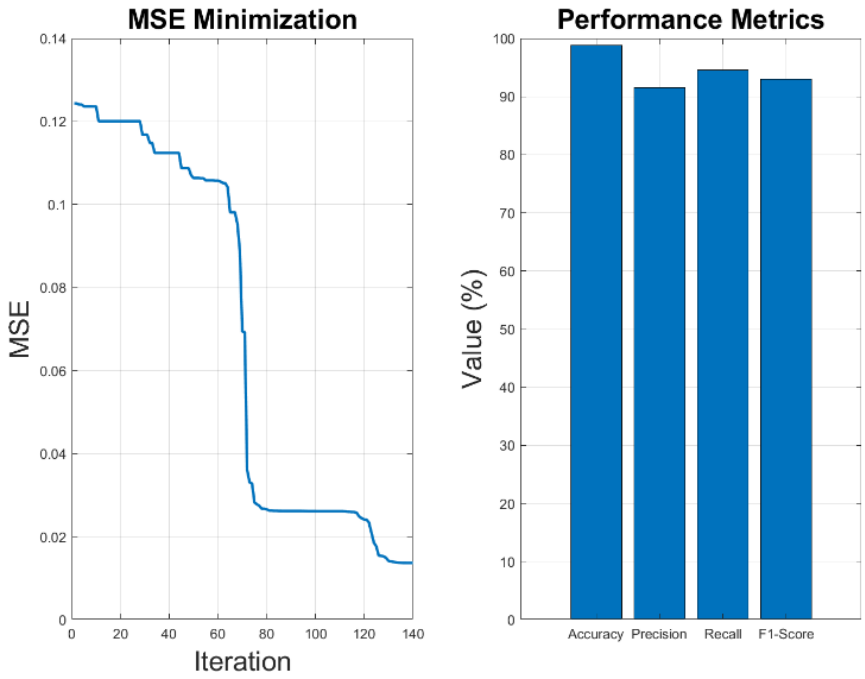


Table 12 presents the results obtained from the proposed method for the simulated CloudArmor dataset.

Table 12

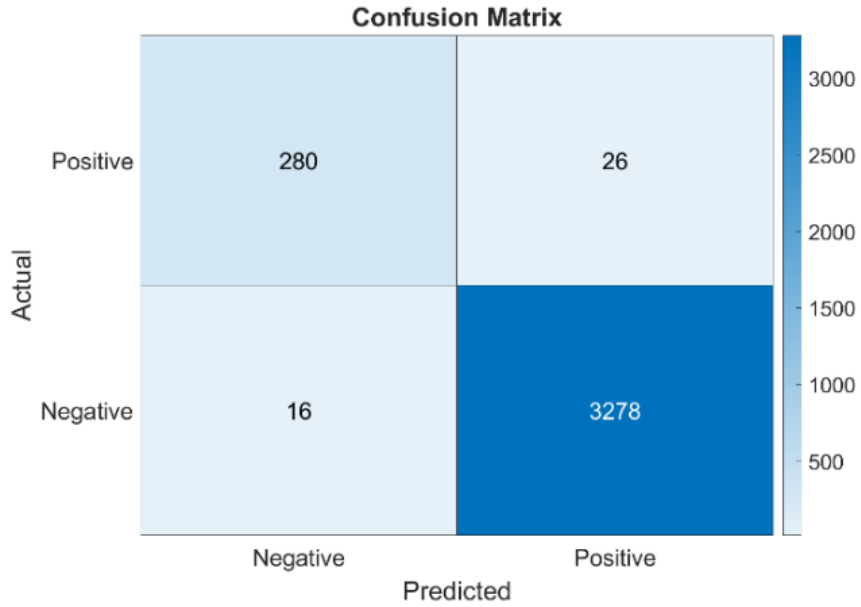
Output of the Particle Swarm Optimization-Based Extreme Learning Machine According to the Evaluation Criteria for the Simulated CloudArmor Dataset

Evaluation Criteria	MSE	RMSE	Accuracy	Precision	Recall	F1-score
Simulated CloudArmor dataset	0.0137	0.1171	98.83%	91.50%	94.59%	93.02%

Based on Figure 11 and Table 12, in the simulated CloudArmor dataset, the proposed model achieved an overall accuracy of 98.83%, indicating its strong capability in correctly identifying samples. In addition, the F1-score of 93.02%and the recall of 94.59%indicate an appropriate balance between precision and recall. The low RMSE and MSE values also indicate suitable convergence and low model error in the prediction process. Overall, these results confirm the reliable and stable performance of the model in detecting fake feedback in cloud environments. Figure 12 shows the confusion matrix of the results obtained by the proposed method on the simulated CloudArmor dataset.

Figure 12

Confusion matrix for the simulated CloudArmor dataset



According to Figure 12 and Table 12, the confusion matrix of the proposed model on the simulated CloudArmor dataset indicates highly satisfactory performance. The confusion matrix presents the results obtained from 30% of the test data, corresponding to approximately 3,600 samples out of a total of 12,000 samples, where the model correctly identified 3,278 true negative samples and 280 true positive samples, while only 26 positive cases were incorrectly predicted as negative and 16 negative samples were incorrectly identified as positive. These results demonstrate the model's performance in distinguishing fake feedback from genuine feedback. The distribution of values in the matrix shows that the model achieved an accuracy of 98.83%and was able to learn the data pattern correctly with minimal classification error. This confirms its high effectiveness in identifying fake feedback and improving prediction quality in cloud environments.

To demonstrate the performance of the proposed method with ELM optimized by PSO, the datasets were prepared using the same AHC-RSA clustering, E-EDA, and labeling procedure, and other implemented machine learning algorithms were then applied. The results are presented in Table 13.

Table 13

Comparison of the Performance of the Proposed Method with Other Machine Learning Algorithms

Machine Learning Algorithm	Evaluation Criteria	Epinions Dataset	Simulated CloudArmor Dataset
Decision Tree (NumberTree = 100)	MSE	0.032199	0.04111
	RMSE	0.17944	0.20276
	Accuracy	96.78%	95.88%
	Precision	100%	65.74%
	Recall	55.32%	100%
	F1-score	71.66%	79.33%
KNN ($K = 3$)	MSE	0.023951	0.036389
	RMSE	0.15476	0.19076
	Accuracy	97.605%	96.36%
	Precision	80.043%	69.97%
	Recall	89.064%	94.36%
	F1-score	84.313%	80.36%
SVM (RBF/Gaussian)	MSE	0.17394	0.057222
	RMSE	0.13189	0.23921
	Accuracy	98.261%	94.27%

	Precision	100%	57.95%
	Recall	75.931%	100%
	F1-score	86.318%	73.38%
ANN	MSE	0.043622	0.036111
	RMSE	0.20886	0.19003
	Accuracy	95.63%	96.38%
	Precision	100%	67.82%
	Recall	40.18%	100%
	F1-score	57.32%	89.98%
LSTM (Number of Neurons = 100)	MSE	0.13425	0.021111
	RMSE	0.11587	0.1453
	Accuracy	98.658%	97.88%
	Precision	100%	78.88%
	Recall	81.423%	100%
	F1-score	89.76%	88.19%
GRU (Number of Neurons = 40)	MSE	0.0075235	0.019722
	RMSE	0.086738	0.14044
	Accuracy	99.24%	98.02%
	Precision	100%	80.00%
	Recall	89.67%	100%
	F1-score	94.55%	88.88%
Proposed Model (ELM-PSO)	MSE	0.0036	0.0137
	RMSE	0.0604	0.1711
	Accuracy	99.69%	98.83%
	Precision	95.82%	91.50%
	Recall	100%	94.59%
	F1-score	97.87%	93.02%

As shown in Table 13, different machine learning algorithms exhibited different levels of performance on the Epinions and CloudArmor datasets. Classical algorithms such as Decision Tree and KNN achieved relatively high accuracy at around 96%, but due to lower recall in the Epinions dataset (55.32%and 89.06%, respectively), they performed more weakly in detecting fake feedback. In contrast, SVM, with an accuracy of approximately 98%and a recall of 75.93%, delivered a more balanced performance, but exhibited greater fluctuation in MSE, indicating its sensitivity to data variation. Among the neural networks, ANN achieved high accuracy, but its recall was lower than that of the other models, suggesting a tendency toward overfitting. The LSTM and GRU algorithms, due to their capability to learn temporal dependencies, performed better than the earlier models and maintained accuracy above 98%. Overall, the proposed algorithm (ELM-PSO), while preserving high stability and the lowest error, outperformed the other models, particularly in reducing MSE and RMSE and in achieving a desirable balance between precision and recall.

3.5. Comparison with Other Models

Due to the limited application of machine learning algorithms in cloud computing for this specific domain, and because previous studies have used different datasets, the comparison of the obtained results with other studies is presented in Table 14. Two studies by Seyadat et al. (Siadat et al., 2017) and Soleymani et al. (Soleymani et al., 2021) used the Epinions dataset; however, neither study employed machine learning algorithms nor explicitly reported evaluation metrics comparable to those used in the present research. In the study by Seyadat et al. (Siadat et al., 2017), a Bayesian game model was applied to analyze and identify malicious users and prevent the submission of their feedback. Simulation results showed that this model could correctly identify malicious users and detect fake feedback. In that study, results were reported based on false negative and false positive measures, where the false negative rate was 0.60and the false positive rate was 0.20. These results indicate that the difference between the trust distribution before and after the injection of fake feedback was 60%, which was reduced to 5% after applying the Bayesian game model, corresponding to a 55% improvement in accuracy and trust distribution reliability. Based on this comparison, the proposed model in this study demonstrates a substantial improvement in accuracy. In the study by Soleymani et al. (Soleymani et al., 2021), which focused on a trust model using fuzzy rules, it was shown that the average confidence and trust level decreased by approximately 29%, corresponding to a 71% accuracy in detection. However,

because these two studies did not explicitly report evaluation criteria consistent with those used in this research, they could not be directly included in the comparative table. Nevertheless, based on the reported false positive and false negative measures in Seyadat et al. (Siadat et al., 2017) and Soleymani et al. (Soleymani et al., 2021), the proposed method in this study performs better according to the confusion matrix results.

Table 14

Comparison of the Average Performance of the Proposed Method with Other Studies

Source	Dataset	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
(Deshai & Rao, 2023)	Amazon	CNN-PSO	99.4%	98.53%	98.01%	99.02%
(Taneja & Kaur, 2021)	CloudArmor	Ensemble model	97.51%	98.19%	93.65%	95.86%
(Alsubari et al., 2022)	TripAdvisor	Random Forest	95%	—	—	—
(Martens & Maalej, 2019)	Apple App	MLP	91%	—	—	—
(Javed et al., 2021)	Yelp	Deep learning ensemble	—	—	—	92%
(Thirunavukkarasu et al., 2023)	Yelp	Logistic Regression	88.43%	—	—	—
(Cho et al., 2023)	Selenium	SVC-KMeans	85%	—	—	—
Proposed method	Epinions	AHC-RSA_E-EDA_ELM-PSO	99.84%	98.84%	100%	99.36%
Proposed method	Simulated CloudArmor	AHC-RSA_E-EDA_ELM-PSO	98.83%	91.50%	94.59%	93.02%

Based on the results presented in Table 14, the comparison between the proposed model and previous studies indicates that the hybrid model AHC-RSA_E-EDA_ELM-PSOprovides superior and more stable performance compared with conventional methods in detecting fake feedback. While reference models such as CNN-PSO achieved an accuracy of 99.4%and an F1-score of 99.02%on the Amazon dataset, and ensemble models on the CloudArmor dataset achieved an accuracy of 97.51%, the proposed model achieved an accuracy of 99.84%and an F1-score of 99.36%on the Epinions dataset, representing the highest level of accuracy and recall. Furthermore, on the simulated CloudArmor dataset, the proposed model achieved an accuracy of 98.83%, demonstrating strong generalization capability across different data environments. In comparison, more classical models such as Logistic Regression and SVC-KMeans achieved accuracies of 88.43%and 85%, respectively, indicating their limitations in extracting complex patterns. Overall, the results indicate that the integration of hierarchical clustering, Reptile Search optimization, and PSO-optimized Extreme Learning Machine significantly improves the accuracy, recall, and stability of the proposed model compared with other approaches. Moreover, the results show that the proposed model AHC-RSA_E-EDA_ELM-PSOachieves a substantial improvement in both accuracy and stability compared with all previous methods. In the Epinions dataset, the improvement in F1-score and accuracy compared with the CNN-PSO model corresponds to approximately 0.8%in overall accuracy and 0.58%in F1-score. Similarly, in the CloudArmor dataset, the proposed model demonstrates improved performance compared with the best existing ensemble model in terms of accuracy. These results indicate that the combination of agglomerative hierarchical clustering, Reptile Search optimization, and PSO-optimized Extreme Learning Machine enhances detection accuracy and reduces classification error compared with previous models.

4. Discussion and Conclusion

The findings of this study demonstrate that the proposed hybrid framework AHC-RSA_E-EDA_ELM-PSO achieves a high level of performance in detecting fake feedback within cloud environments, as evidenced by the consistently strong evaluation metrics across both the Epinions and simulated CloudArmor datasets. Specifically, the model attained accuracy values exceeding 98%, recall values approaching or reaching 100%, and high F1-scores, indicating a well-balanced trade-off between precision and recall. These results confirm that the integration of hierarchical clustering, enhanced empirical data analytics, and optimized machine learning provides a robust mechanism for identifying fraudulent feedback patterns. The low values of MSEand RMSEfurther indicate that the model converges effectively and maintains minimal prediction error, suggesting strong generalization capabilities in complex and noisy data environments.

The superior performance of the proposed model can be attributed to its multi-stage architecture, where each component contributes to improving the overall detection process. The initial application of improved agglomerative hierarchical clustering using the Reptile Search Algorithm (RSA) plays a critical role in reducing data dispersion and addressing the cold-start problem. By grouping similar users and feedback into clusters, the model enhances the separability of data, thereby facilitating more accurate anomaly detection. This finding is consistent with previous studies that emphasize the effectiveness of clustering techniques in uncovering hidden structures in large datasets and improving anomaly detection performance (Shi et al., 2020; Zhang et al., 2020). Furthermore, the use of RSA for optimizing feature weights introduces an adaptive mechanism that enhances clustering quality, aligning with research highlighting the importance of optimization algorithms in improving model accuracy and convergence (Albadr et al., 2024; Maheshwari & Begde, 2025).

The second component of the proposed framework, namely the enhanced empirical data analytics (E-EDA) method, significantly contributes to the accurate identification of fake feedback by leveraging statistical properties of the data. The use of cumulative proximity, unimodal and multimodal density measures, and Chebyshev's inequality enables the model to detect both global and local anomalies without relying on labeled data. This approach is particularly effective in handling imbalanced datasets and sparse data distributions, which are common challenges in cloud-based feedback systems. The results of this study support prior findings that statistical and density-based anomaly detection methods can effectively identify outliers and fraudulent patterns in complex datasets (Gu, 2017; Salminen et al., 2022). Moreover, the integration of frequency-based weighting in the multimodal density calculation enhances the model's ability to distinguish low-frequency anomalous patterns, which is consistent with research emphasizing the importance of frequency and behavioral analysis in detecting deceptive reviews (Birim et al., 2022; He et al., 2022).

The final stage of the proposed framework involves classification using an Extreme Learning Machine optimized by Particle Swarm Optimization (ELM-PSO), which demonstrates superior predictive performance compared with conventional machine learning and deep learning models. The optimization of input weights and hidden-layer biases using PSO addresses the instability associated with random initialization in ELM, resulting in improved convergence and classification accuracy. This finding aligns with previous studies that highlight the effectiveness of PSO in enhancing the performance of neural network models by optimizing their parameters (Albadr et al., 2024; Maheshwari & Begde, 2025). Additionally, the use of ELM as the classification model provides computational efficiency and rapid training, making it suitable for large-scale cloud environments where real-time processing is required (Yu et al., 2015). Compared with deep learning approaches such as CNN, LSTM, and GRU, the proposed model achieves comparable or superior performance with lower computational complexity, which is a significant advantage in practical applications (Deshai & Rao, 2023; Javed et al., 2021).

The comparative analysis with other machine learning algorithms further reinforces the effectiveness of the proposed approach. While traditional models such as Decision Tree and KNN achieved moderate accuracy, their lower recall values indicate limited capability in detecting all instances of fake feedback. This limitation is consistent with findings in previous studies that highlight the challenges faced by classical machine learning models in handling imbalanced datasets and complex patterns (Ram et al., 2022; Thirunavukkarasu et al., 2023). In contrast, the proposed model achieves a recall of up to 100%, indicating its ability to identify all fraudulent instances. Similarly, although deep learning models such as LSTM and GRU demonstrated high accuracy, their performance is often accompanied by increased computational cost and sensitivity to hyperparameter tuning (Hajek et al., 2020; Manaskasemsak et al., 2023). The proposed model overcomes these limitations by combining efficient learning mechanisms with optimization techniques, resulting in a more balanced and scalable solution.

Another important observation from the results is the effectiveness of incorporating clustering-based feature augmentation. By assigning cluster labels as additional features, the model captures higher-level patterns of user behavior and feedback characteristics. This approach enhances the predictive capability of the classifier by providing contextual information that is not directly available in the original feature space. This finding is consistent with studies that emphasize the role of feature engineering and representation learning in improving classification performance (Barbado et al., 2019; Cho et al., 2023). Furthermore, the ability of the model to maintain high performance across different datasets, including a simulated CloudArmor dataset, demonstrates its robustness and

generalizability. This is particularly important in cloud environments, where data characteristics may vary significantly across different applications and domains (Guo et al., 2023; Islam et al., 2023).

The statistical validation of the results using ANOVA further confirms the reliability and stability of the proposed model. The significant *p*-values indicate that the observed improvements in performance are not due to random variation but are statistically meaningful. This aligns with prior research emphasizing the importance of rigorous statistical validation in evaluating machine learning models (Ennaouri & Ahmed, 2022). Additionally, the low variance observed in evaluation metrics across multiple runs suggests that the model is not sensitive to initialization or data partitioning, which is a desirable property for real-world deployment.

Despite these promising results, it is important to consider the broader context of fake feedback detection research. Previous studies have explored various approaches, including Bayesian game models, fuzzy rule-based systems, and network-based methods, each with its own strengths and limitations (Goswami et al., 2017; Siadat et al., 2017; Soleymani et al., 2021). While these methods have contributed to advancing the field, they often lack the ability to integrate multiple sources of information and adapt to dynamic environments. The proposed hybrid framework addresses these limitations by combining complementary techniques into a unified model, thereby enhancing its overall effectiveness.

Overall, the results of this study demonstrate that the integration of improved hierarchical clustering, enhanced statistical analysis, and optimized machine learning provides a powerful and efficient solution for fake feedback detection in cloud environments. The proposed model not only achieves high accuracy and stability but also offers scalability and adaptability, making it suitable for practical applications in real-world cloud systems.

One of the primary limitations of this study is the reliance on a simulated version of the CloudArmor dataset due to the unavailability of the original dataset. Although the synthetic dataset was carefully designed to replicate the characteristics of real-world data, it may not fully capture all the complexities and variations present in actual cloud environments. Additionally, the labeling of fake feedback in the dataset is based on predefined rules and assumptions, which may introduce bias into the evaluation process. Another limitation is the computational complexity associated with the hierarchical clustering component, particularly for large-scale datasets, which may affect the scalability of the model. Furthermore, the model focuses primarily on structured and semi-structured data, and the integration of more advanced natural language processing techniques for analyzing unstructured textual feedback remains limited.

Future research can address these limitations by utilizing real-world datasets and exploring more diverse data sources to improve the robustness and generalizability of the model. The integration of advanced deep learning techniques, such as transformer-based models, could further enhance the analysis of textual feedback and improve detection accuracy. Additionally, developing more efficient clustering algorithms or employing approximate methods could reduce computational complexity and improve scalability. Future studies may also investigate the use of explainable artificial intelligence techniques to enhance the interpretability of the model and provide insights into the decision-making process. Moreover, incorporating temporal and network-based features could enable the detection of coordinated and evolving fraudulent behaviors, further improving the effectiveness of the model.

From a practical perspective, the proposed framework can be implemented in cloud service platforms to enhance trust management systems and improve the reliability of user feedback. Organizations can use this model to automatically detect and filter fake feedback, thereby ensuring that users receive accurate and trustworthy information when selecting services. The model can also be integrated into recommendation systems to improve the quality of service suggestions and enhance user satisfaction. Furthermore, policymakers and platform administrators can leverage the insights generated by the model to develop more effective strategies for preventing fraudulent activities and maintaining the integrity of online ecosystems.

Authors' Contributions

Authors contributed equally to this article.

Declaration

In order to correct and improve the academic writing of our paper, we have used the language model ChatGPT.

Transparency Statement

Data are available for research purposes upon reasonable request to the corresponding author.

Acknowledgments

We would like to express our gratitude to all individuals helped us to do the project.

Declaration of Interest

The authors report no conflict of interest.

Funding

According to the authors, this article has no financial support.

Ethics Considerations

In this research, ethical standards including obtaining informed consent, ensuring privacy and confidentiality were considered.

References

Aghaee Ghazvini, G., Mohsenzadeh, M., Nasiri, R., & Rahmani, A. M. (2020). A New Multi-Level Trust Management Framework (MLTM) for Solving the Invalidity and Sparse Problems of User Feedback Ratings in Cloud Environments. *The Journal of Supercomputing*, 1-31. https://doi.org/10.1007/s11227-020-03348-1

Ahmed, H., Traore, I., & Saad, S. (2018). Detecting Opinion Spams and Fake News Using Text Classification. *Security and Privacy*, *1*(1), 1-15. https://doi.org/10.1002/spy2.9

Albadr, M. A. A., Tiun, S., Ayob, M., & Al-Dhief, F. T. (2024). Particle Swarm Optimization-Based Extreme Learning Machine for COVID-19 Detection. *Cognitive Computation*, *16*, 1858-1873. https://doi.org/10.1007/s12559-022-10063-x

Alsubari, S. N., Deshmukh, S. N., Alqarni, A. A., Alsharif, N., Aldhyani, H. H. T., Alsaade, F. W., & Khalaf, O. I. (2022). Data Analytics for the Identification of Fake Reviews Using Supervised Learning. *Computers, Materials & Continua*, *70*(2), 3189-3204. https://doi.org/10.32604/cmc.2022.019625

Asaad, W. H., Allami, R., & Yossra, H. A. (2023). Fake Review Detection Using Machine Learning. *Revue d'Intelligence Artificielle*, *37*(5), 1159-1166. https://doi.org/10.18280/ria.370507

Barbado, R., Araque, O., & Iglesias, C. A. (2019). A Framework for Fake Review Detection in Online Consumer Electronics Retailers. *Information Processing & Management*, *56*(4), 1234-1244. https://doi.org/10.1016/j.ipm.2019.03.002

Birim, Ş. Ö., Kazancoglu, I., Kumar Mangla, S., Kahraman, A., Kumar, S., & Kazancoglu, Y. (2022). Detecting Fake Reviews Through Topic Modelling. *Journal of Business Research*, *149*, 884-900. https://doi.org/10.1016/j.jbusres.2022.05.081

Cho, W., Nam, K., Park, M., Yang, S., Hwang, S., & Oh, H. (2023). Fake Review Identification and Utility Evaluation Model Using Machine Learning. *Frontiers in Artificial Intelligence*, *5*, 1064371. https://doi.org/10.3389/frai.2022.1064371

Deshai, N., & Rao, B. B. (2023). Unmasking Deception: A CNN and Adaptive PSO Approach to Detecting Fake Online Reviews. *Soft Computing*, 1-22. https://doi.org/10.1007/s00500-023-08507-z

Ennaouri, M., & Ahmed, Z. (2022). Fake Reviews Detection Through Machine Learning Algorithms: A Systematic Literature Review. https://doi.org/10.21203/rs.3.rs-2039197/v1

Goswami, K., Park, Y., & Song, C. (2017). Impact of Reviewer Social Interaction on Online Consumer Review Fraud Detection. *Journal of Big Data*, *4*(1), 1-19. https://doi.org/10.1186/s40537-017-0075-6

Gu, X. (2017). *Autonomous Anomaly Detection* 2017 Evolving and Adaptive Intelligent Systems (EAIS), https://doi.org/10.1109/EAIS.2017.7954831

Guo, R., Tafti, A., & Subramanyam, R. (2023). Internal IT Modularity, Firm Size, and Adoption of Cloud Computing. *Electronic Commerce Research*, 1-30. https://doi.org/10.1007/s10660-023-09691-8

Hajek, P., Barushka, A., & Munk, M. (2020). Fake Consumer Review Detection Using Deep Neural Networks Integrating Word Embeddings and Emotion Mining. *Neural Computing and Applications*, *32*(23), 17259-17274. https://doi.org/10.1007/s00521-020-04757-2

He, S., Ollenbeck, B., Overgoor, G., Proserpio, D., & Osyali, A. (2022). Detecting Fake-Review Buyers Using Network Structure: Direct Evidence from Amazon. *Proceedings of the National Academy of Sciences*, *119*(47), 1-5. https://doi.org/10.1073/pnas.2211932119

Islam, R., Patamsetti, V., Gadhi, A., Madhavi Gondu, R., Bandaru, C. M., Kesani, S. C., & Abiona, O. (2023). The Future of Cloud Computing: Benefits and Challenges. *International Journal of Communications, Network and System Sciences*, *16*(4), 53-65. https://doi.org/10.4236/ijcns.2023.164004

Jagpreet, S., & Sarbjeet, S. (2017). Improved TOPSIS Method Based Trust Evaluation Framework for Determining Trustworthiness of Cloud Service Providers. *Journal of Grid Computing*, *15*, 81-105. https://doi.org/10.1007/s10723-016-9363-1

Javed, M. S., Majeed, H., Mujtaba, H., & Beg, M. O. (2021). Fake Reviews Classification Using Deep Learning Ensemble of Shallow Convolutions. *Journal of Computational Social Science*, *4*(1), 883-902. https://doi.org/10.1007/s42001-021-00114-y

Liu, W., He, J., Han, S., & Zhu, N. (2019). *A Method for the Detection of Fake Reviews Based on Temporal Features of Reviews and Comments* 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019): Advances in Computer Science Research, https://doi.org/10.2991/icmeit-19.2019.97

Liu, Y., & Pang, B. (2018). A Unified Framework for Detecting Author Spamicity by Modeling Review Deviation. *Expert Systems with Applications*, *112*, 148-155. https://doi.org/10.1016/j.eswa.2018.06.028

Maheshwari, M., & Begde, P. (2025). The role of artificial intelligence and machine learning in enhancing tax compliance and fraud detection in India. *Journal of Informatics Education and Research*, *5*(2).

Manaskasemsak, B., Tantisuwankul, J., & Rungsawang, A. (2023). Fake Review and Reviewer Detection Through Behavioral Graph Partitioning Integrating Deep Neural Network. *Neural Computing and Applications*, *35*, 1169-1182. https://doi.org/10.1007/s00521-021-05948-1

Martens, D., & Maalej, W. (2019). Towards Understanding and Detecting Fake Reviews in App Stores. *Empirical Software Engineering*, *24*, 3316-3355. https://doi.org/10.1007/s10664-019-09706-9

Miradi, H., Keshavarz Saheepoush, M., & Mir Abadini, S. J. (2025). Malware Detection Using Machine Learning and Deep Learning Models: A Systematic Review. Tehran.

Mujawar, T. N., & Bhajantri, L. B. (2022). Behavior and Feedback Based Trust Computation in Cloud Environment. *Journal of King Saud University - Computer and Information Sciences*, *34*(8), 4956-4967. https://doi.org/10.1016/j.jksuci.2020.12.003

Ram, N. C. S., Vakati, G., Nadimpalli, J. V., Sah, Y., & Datla, S. K. (2022). Fake Reviews Detection Using Supervised Machine Learning. https://doi.org/10.22214/ijraset.2022.43202

Salminen, J., Kandpal, C., Kamel, A. M., Jung, S., & Jansen, B. J. (2022). Creating and Detecting Fake Reviews of Online Products. *Journal of Retailing and Consumer Services*, *64*, 102771. https://doi.org/10.1016/j.jretconser.2021.102771

Shi, P., Zhao, Z., Zhong, H., Shen, H., & Ding, L. (2020). An Improved Agglomerative Hierarchical Clustering Anomaly Detection Method for Scientific Data. *Concurrency and Computation: Practice and Experience*. https://doi.org/10.1002/cpe.6077

Shilpa, D., & Rajesh, I. (2018). Evidence Based Trust Estimation Model for Cloud Computing Services. *International Journal of Network Security*, *20*(2), 291-303.

Siadat, S., Rahmani, A. M., & Navid, H. (2017). Identifying Fake Feedback in Cloud Trust Management Systems Using Feedback Evaluation Component and Bayesian Game Model. *The Journal of Supercomputing*, *73*(6), 2682-2704. https://doi.org/10.1007/s11227-016-1950-1

Soleymani, M., Abapour, N., Taghizadeh, E., Siadat, S., & Karkehabadi, R. (2021). Fuzzy Rule-Based Trust Management Model for the Security of Cloud Computing. *Mathematical Problems in Engineering*, 1-14. https://doi.org/10.1155/2021/6629449

Taneja, H., & Kaur, S. (2021). An Ensemble Classification Model for Fake Feedback Detection Using Proposed Labeled CloudArmor Dataset. *Computers & Electrical Engineering*, *93*(1), 1-15. https://doi.org/10.1016/j.compeleceng.2021.107217

Thirunavukkarasu, M., Kavya, P., & Mahalakshmi, P. T. (2023). Fake Reviews Detection Using Supervised Machine Learning. *International Journal of Engineering Research and Applications*, *13*(4), 250-254. https://doi.org/10.9790/9622-1304250254

Truong, X., Nguyen, T. T., Tran, V. K., Quach, S., Thaichon, P., Jo, J., Vo, B., Tran, Q. D., & Nguyen, Q. V. H. (2023). Towards a Review-Analytics-as-a-Service (RAaaS) Framework for SMEs: A Case Study on Review Fraud Detection and Understanding. *Australasian Marketing Journal*, *32*(1), 76-90. https://doi.org/10.1177/14413582221146004

Wang, Z., Hu, R., Chen, Q., Gao, P., & Xu, X. (2020). Collueagle: Collusive Review Spammer Detection Using Markov Random Fields. *Data Mining and Knowledge Discovery*, *34*(6), 1621-1641. https://doi.org/10.1007/s10618-020-00693-w

Ye, J., & Akoglu, L. (2015). *Discovering Opinion Spammer Groups by Network Footprints* Joint European Conference on Machine Learning and Knowledge Discovery in Databases, https://doi.org/10.1007/978-3-319-23528-8_17

Yu, L., Danninga, Z., & Hongbinga, C. (2015). Prediction of Length-of-Day Using Extreme Learning Machine. *Geodesy and Geodynamics*, *16*(2), 151-159. https://doi.org/10.1016/j.geog.2014.12.007

Zhang, F., Hao, X., Chao, J., & Yuan, S. (2020). Label Propagation-Based Approach for Detecting Review Spammer Groups on E-Commerce Websites. *Knowledge-Based Systems*, *193*, 105520. https://doi.org/10.1016/j.knosys.2020.105520

Zhang, W., Xie, R., Wang, Q., Yang, Y., & Li, J. (2022). A Novel Approach for Fraudulent Reviewer Detection Based on Weighted Topic Modelling and Nearest Neighbors with Asymmetric Kullback-Leibler Divergence. *Decision Support Systems*, *157*, 113765. https://doi.org/10.1016/j.dss.2022.113765