Power System Protection and Control
ISSN:1674-3415

PRESENTING IMPROVED A COMBINED STATISTICAL/LEARNING APPROACH TO DETECT FAKE FEEDBACK IN CLOUD ENVIRONMENT USING HIERARCHICAL CLUSTERING WITH RSA AND EXTREME LEARNING MACHINES WITH PSO

Hussein Kadhim Almamoori[1], Golnaz Aghaee Ghazvini[2], Ali Albu-Rghaif[3], Fariba Majidi[4]

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

Corespondance Author: Golnaz Aghaee Ghazvini
G.aghaee@iau.ac.ir

Received: 27/12/2025 **Revised**: 26/01/2026 **Accepted**: 22/02/2026

***ABSTRACT*:**

The increase in fake feedback in cloud environments is one of the main challenges in maintaining user trust and ensuring service quality. Therefore, developing intelligent approaches to distinguish real feedback from fake has become increasingly important. In this study, a combined framework based on statistical analysis and machine learning is presented for fake feedback detection, in which the improved hierarchical clustering with reptile search algorithm (AHC-RSA), the E-EDA statistical model for anomaly filtering, and the optimized extreme learning model with particle swarm optimization (ELM-PSO) are integrated. This combination has reduced intra-cluster dispersion, improved data separation, increased prediction accuracy, and accelerated model convergence. Evaluations on two datasets, CloudArmor and Epinions, showed that the proposed model achieved 99.83% and 99.84% accuracy, respectively. The results show that the model is superior to SVM, ANN, LSTM and GRU algorithms in terms of accuracy, stability and convergence speed. On average, the proposed model has about 5 to 8 percent improvement in accuracy and F1-score and a significant reduction in MSE and RMSE errors compared to the best previous methods. As a result, the proposed framework can be used as an effective and reliable method for detecting fake feedback and improving trust in cloud services.

Keywords*: Fake feedback detection, clustering with reptile search algorithm, improved empirical analysis method, optimized extreme learning model with particle swarm algorithm.*

INTRODUCTION

Cloud computing is a paradigm that originated in the 1960s with the development of time-sharing models, aiming to provide applications, frameworks, and infrastructure resources as services [1, 2]. It delivers computing as a service and serves as an accessible, scalable, and flexible computational platform for a wide range of applications [3, 4]. The primary advantage of cloud computing lies in its ability to allocate resources dynamically based on user demand and to offer a pay-as-you-go pricing model [5]. Despite its numerous benefits, several challenges persist. One of the most critical is the collection and quality of user feedback [2, 3]. In recent years, feedback and review-sharing platforms have grown significantly, playing an increasingly vital role in shaping the decisions of potential users. However, due to the prevalence of fake or misleading feedback, cloud service providers often face issues such as reduced user trust, financial losses, and reputational damage [6, 7].

Another significant challenge is the **cold-start problem** and the **dispersion of feedback.** Many new users lack prior interaction experience with service providers and thus do not submit feedback, resulting in insufficient data for effective analysis and fake feedback detection [8].

To address these issues, various approaches have been proposed for **fake feedback detection**, which generally fall under the domains of **Natural Language Processing (NLP)** and **sentiment analysis** [9,10]. These systems aim to analyze and distinguish between genuine and deceptive feedback. However, most existing studies rely on **supervised learning models**, which require labeled datasets—often generated synthetically or through manual annotation. Commonly used algorithms in this domain include **Support Vector Machines (SVM) with RBF kernels, k-Nearest Neighbors (k-NN), Neural Networks**, and **Bayesian Networks**. In contrast, **unsupervised learning approaches** have been less frequently explored, despite their advantage of not requiring labeled data [6]. Overall, the main challenges addressed in this research are the **user cold- start problem, feedback sparsity, limited availability of labeled datasets**, and **data quality concerns** [6,8,9].

To address the challenges outlined above, the aim of this research is to present a combined statistical/learning approach for detecting fake feedback in the cloud environment using improved hierarchical clustering and ELM optimized with PSO algorithm. Because the challenge of cold start and data dispersion can be overcome by using clustering, labeling and data quality of the dataset can be addressed using statistical methods and then the accuracy of the model can be measured using extreme learning machine algorithms. The main contributions of this paper are as follows:

1- Presenting an innovative method combining aggregate hierarchical clustering and RSA algorithm to optimize feature weights and select appropriate distance in clustering: Presenting an innovative method combining AHC and RSA algorithm that reduces intra-cluster dispersion by optimizing feature weights and selecting appropriate distance, solves the problem of unbalanced clusters and shard initiation, improves the quality and resolution of clusters, and enables accurate clustering of unlabeled data.
2- The proposed advanced E-EDA helps to improve anomaly detection in non-normal data distribution by integrating dual-scale density estimation, frequency-weighted multimodal analysis, and Chebyshev-based anomaly filter.
3- Presenting a combined statistical/learning approach: Using a fully automated method for detecting fake feedback, labeling the dataset using a statistical method, and an ELM-PSO algorithm. Meta-heuristic methods can be used to increase the accuracy of the extreme vector machine. In this research, a PSO algorithm can be presented for a compact network architecture with better performance.
4- Implementation of the proposed method: This method is implemented on two valid Rich Epinions datasets and a simulated CloudArmor dataset.
This paper is organized into five sections. The first section introduces fake feedback and its challenges. The second section reviews related studies. The third section presents the proposed combined statistical/learning method and its algorithms. The fourth section analyzes the implementation, evaluates performance, and compares results with existing methods. Finally, the fifth section presents conclusions and suggestions for future research.

RELATED WORK

Previous studies on fake feedback detection can be broadly categorized into two main areas:
1. Fake Feedback Detection in Cloud Environments: In the cloud computing domain, research has primarily focused on identifying fake feedback in trust management systems and improving service quality. For example: Siadat et al. [2] employed a Bayesian game model and feedback evaluation components to detect malicious users, demonstrating the model's effectiveness in identifying harmful actors. Mujawar & Bhajantri [3] proposed behavior- and feedback-based trust computations, showing that their method outperformed existing evidence- and service-quality-based models in terms of accuracy. Other studies include a cloud certificate-based trust model [11], a compliance-monitoring framework [12], and an evidence-based trust estimation model [13]. Taneja & Kaur [5] introduced a group classification model for fake feedback detection using labeled data from CloudArmor, showing improved performance over supervised classifiers such as nearest neighbor, logistic regression, and SVM. Additional approaches in this domain include unsupervised and clustering-based techniques [14, 15], graph-based models [16–19], and feature modeling approaches [20, 21]. Furthermore, multi-level trust management frameworks [7, 8] and supervised learning algorithms [9, 22–26] have been applied to detect fake feedback, demonstrating improved feedback quality and provider trustworthiness.

2. Fake Feedback Detection in Other Domains: Outside cloud environments, studies have focused on machine learning and deep learning techniques to detect fake feedback: Deshai & Rao [27] applied CNN and PSO optimization for fake feedback detection, achieving better performance than existing models. Other studies employed supervised classification algorithms [28–31], advanced language models such as ULMFiT and GPT-2 [32], and convolutional neural networks [34], achieving high accuracy in identifying fake feedback. Additionally,

Martens & Maalej [35] used MLP neural networks on imbalanced labeled datasets and achieved 91% accuracy, while Liu et al. [36] explored temporal feedback features for detecting fake reviews.

Research Gap: Although previous works have demonstrated the effectiveness of supervised and deep learning methods in detecting fake feedback, several challenges persist. Most existing models rely heavily on labeled data, making them unsuitable for large-scale or dynamic cloud environments where unlabeled feedback dominates. Furthermore, clustering-based methods often face issues of **imbalanced cluster formation, distance metric selection**, and **feature-weight optimization**, leading to reduced cluster resolution and misclassification of subtle anomalies. In addition, prior statistical models lack robustness in handling **non-normal data distributions** and **high-dimensional correlations.** Therefore, there is a pressing need for an **integrated statistical-learning framework** capable of optimizing clustering structure, handling non-normality, and automating labeling to improve the accuracy and interpretability of fake feedback detection in cloud environments. Table 1 summarizes some of these studies and their key results.

Table 1. **Summarizes key studies on fake feedback detection using machine learning methods**

Author	**Year**	**Proposed Method**	**Environment**	**Software & Dataset**	**Results**
Deshai & Rao [27]	2023	Combination of Convolutional Neural Network (CNN), PSO, and Natural Language Processing (NLP) techniques	Non-cloud computing	Ott, Amazon, Yelp, TripAdvisor, and IMDb datasets; Python software	Accuracy rate exceeding 99.4%.
Thirunavuk Karasu et al. [28]	2023	Machine learning methods: k-Nearest Neighbors (k-NN), Bayesian Network, and Logistic Regression	Non-cloud computing	Yelp dataset; PyCharm software	Accuracy 60.75%, 73.37%, and 88.43%,
Choi et al. [29]	2023	Multi-class Support Vector Classification (SVC) and K-means clustering	Non-cloud computing	Dataset from Selenium; Python software	Accuracy 85%.
Alsubari et al. [9]	2022	Naïve Bayes, Support Vector Machine (SVM), Adaptive Boosting, and Random Forest	Cloud computing	Data collected from TripAdvisor website	Accuracy, precision, recall, and F1-score were 88%, 93%, 94%, and 95%, respectively.
Taneja & Kaur [5]	2021	Cluster-based classification model	Cloud computing	Dataset collected from CloudArmor; Python software	Accuracy, precision, recall, and F1-score were 97.51%, 98.19%, 93.65%, and 95.86%.
Javed et al. [34]	2021	Deep learning ensemble of shallow convolutions and Bag-of-n-grams method	Non-cloud computing	Filtered Yelp dataset; Python software	F1-score 92%
Martens & Maalej [35]	2019	MLP Neural Network	Non-cloud computing	Apple App Store; Python software	Accuracy of 91%.

PROPOSED METHOD

To solve the problem of detecting fake feedback in the cloud environment, the proposed method is applied to detect any anomaly, including outliers and fake data, on the desired dataset. In this study, the user feedback is considered to be about the services they receive from the cloud server. Now, for these services, they can use one

DOI: 10.46121/pspc.54.1.23

or more servers to run their programs. Therefore, considering the statement of the problem, the challenges of this research are: detecting fake feedback in the cloud environment; lack of research in this area; lack of publicly available datasets with appropriate features; lack of labeling of fake feedback classes for the dataset; dispersion of feedbacks. Therefore, the proposed method for detecting fake feedback in this study is based on labeling and statistical analysis of data, and is shown in Figure 1. In the figure, it is indicated as a dashed line according to the innovation and application of the available appropriate algorithms. The steps of the proposed method are as follows:
Step 1: using a standard dataset;
Step 2: preprocessing and sorting the dataset;
Step 3: Using cumulatively improved AHC with RSA algorithm to solve the problem of dispersion and cold start;
Step 4: Analyze the dataset using E-EDA method;
Step 5: Detect fake feedback in the dataset and create class record;
Step 6: Classify with ELM algorithm optimized with PSO algorithm;
Step 7: Estimate the classifier error with evaluation criteria from standard criteria such as MSE, RMSE, Accuracy, Recall, Precision and F1-score.

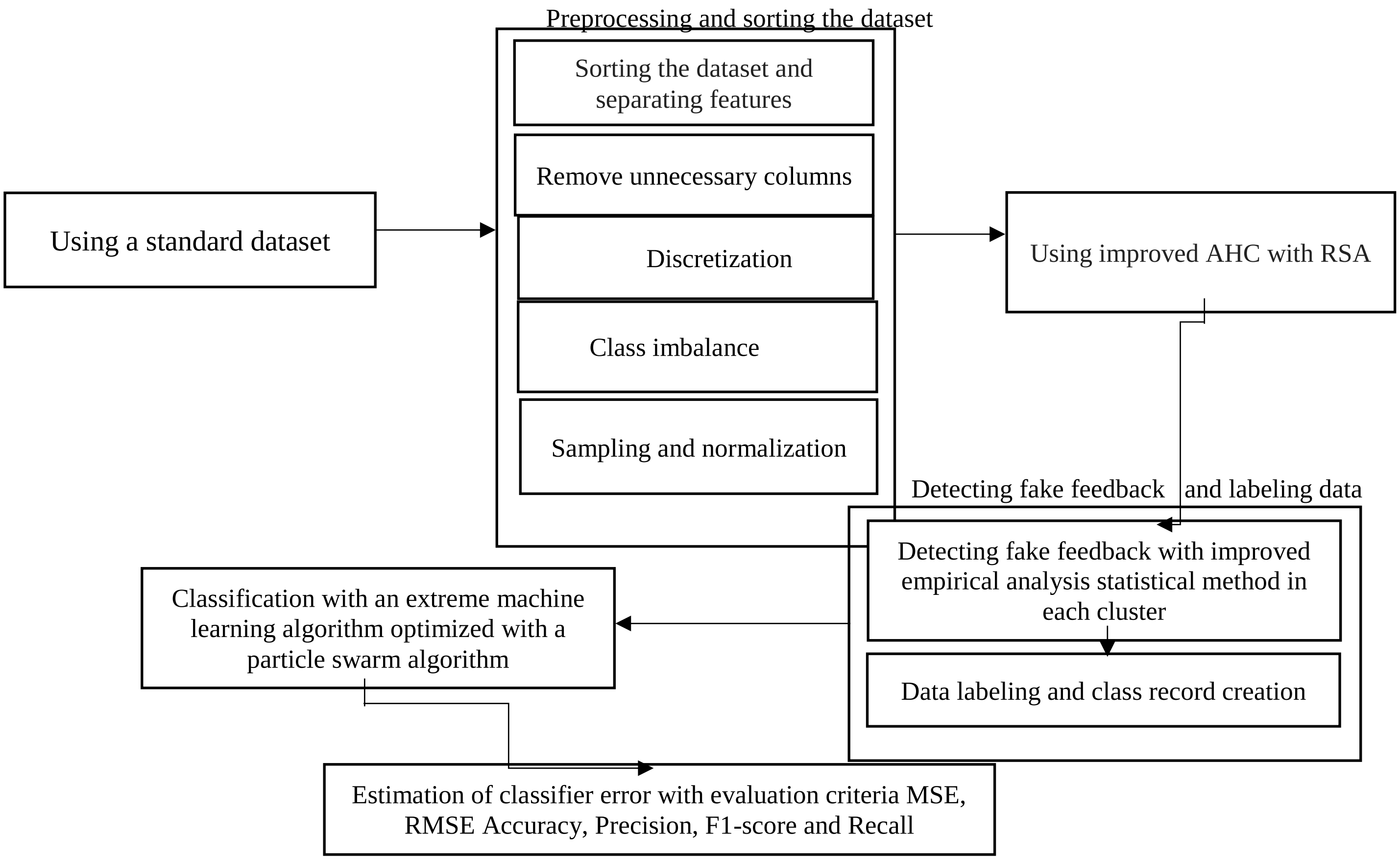
Preprocessing and sorting the dataset
Sorting the dataset and separating features
Remove unnecessary columns
Discretization
Class imbalance
Sampling and normalization
Using a standard dataset
Using improved AHC with RSA
Detecting fake feedback and labeling data
Detecting fake feedback with improved empirical analysis statistical method in each cluster
Data labeling and class record creation
Classification with an extreme machine learning algorithm optimized with a particle swarm algorithm
Estimation of classifier error with evaluation criteria MSE, RMSE Accuracy, Precision, F1-score and Recall

Fig. 1. Block diagram of the proposed method

The algorithms and steps required to present the proposed method are as follows:

3-1- Preprocessing

To use the proposed method, it is necessary to preprocess the data set. For data preprocessing, the following steps are considered, which are given in Figure 2:
1. Sorting the data set and separating the features: To perform the analysis, it is necessary to separate the desired features for each user and specify and sort them in the form of a column. For example, in the data set, the features are stored as (51908,182,4,'Helpful',12223,'2001-02-09') for each user.
2. Removing unnecessary columns: Time columns that are not necessary for the analysis have been removed.
3. Discretization: Text values such as Not Yet Rated are set to 1, Somewhat Helpful to 2, Helpful to 3, Very Helpful to 4, and Show to 5.

4. Class Imbalance: One of the challenges in this study is the small number of detected anomaly data, which are distributed in an imbalanced manner such that the number of anomalies is significantly lower than that of normal data. This imbalance may lead machine learning models to bias toward the majority class and reduce prediction accuracy. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) was employed. By generating Synthetic samples for the minority class (anomaly data), SMOTE balances the data distribution.
5. Sampling and Normalization: To use the ELM algorithm optimized with the PSO algorithm and evaluate it, the data set is divided into two groups: training (70%) and testing (30%). This is done by random sampling. Also, the data is normalized between 0 and 1 before entering the learning algorithm.

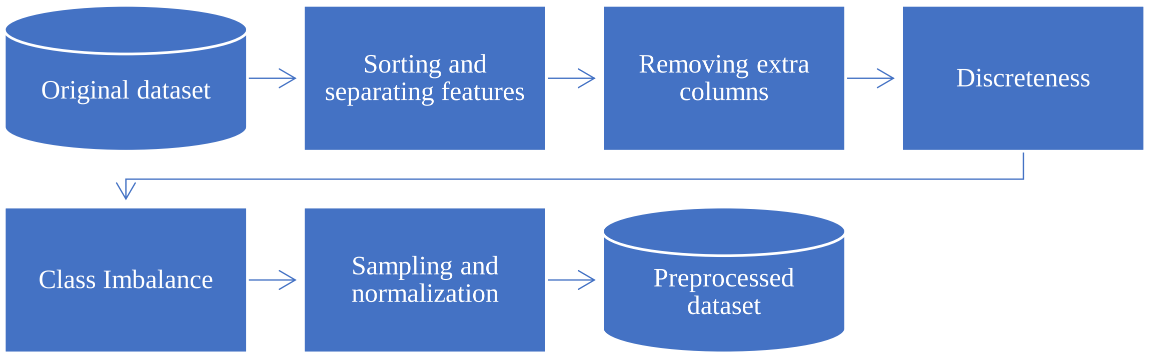

Fig. 2. Preprocessing steps on the dataset

3-2- Improved AHC with RSA
By using clustering and clustering users with common opinions, the dispersion of feedback can be resolved and fake feedback detection can be made easier. Clustering is done with improved AHC. In this method, unlike K-Means clustering, each observation may be placed in more than one cluster because clusters are formed based on different levels of distance. Therefore, each cluster may be a subset of another cluster at a different level of distance. In this method, each data is initially considered as a separate cluster and in an iterative process, at each stage, clusters that are more similar to each other are combined to finally obtain a cluster. The proposed algorithm is a combination of AHC and RSA, which is as follows:

1. **AHC:** AHC is a common unsupervised clustering method that represents the hierarchical structure of data as a tree (dendrogram). In this method, each sample is first considered as a separate cluster, and then at each step, the two clusters that have the most similarity (or the least distance) are merged together. This process continues until all samples are placed in one cluster. The choice of distance criterion and feature weights plays a fundamental role in the quality of clustering. In the classical AHC method, the similarity or distance between samples is calculated based on criteria such as Euclidean distance and then the samples are recursively merged to finally form a hierarchical cluster structure. The main weakness of this method is that it gives equal weight to all features and the choice of distance type is also fixed and predetermined. While in real data, some features are more important than others and choosing the appropriate distance can have a significant impact on the quality of clustering. To overcome this limitation, we use the RSA to optimally select the feature weights and also adjust the distance criterion.

2. **RSA:** The RSA algorithm is a new metaheuristic method introduced by Abualigah et al. in 2022 and modeled on the hunting and locomotion behavior of reptiles in nature. The algorithm is inspired by two key phases in reptile behavior: exploration and exploitation. In the exploration phase, the reptile moves through different environments to discover new areas of the search space, while in the exploitation phase, the reptile focuses on promising positions and makes a final attack to find the best possible position. The algorithm process is described as follows. Let (N) be the number of individuals in the population, (T) be the total number of iterations, and () be the position of the (i)-th c reptile at iteration (t). The search space is bounded by lower and upper bounds () and (), respectively. In the initialization phase, the position of each reptile is randomly determined in the search space, which is represented by Equation 1:

(1)

After initialization, each position is evaluated against the objective function () to determine the best initial position, i.e. (). The RSA algorithm switches between two main phases in each iteration. In the exploration phase, the reptile's movement to explore new areas is modeled as two:

(2)

where () is a random reptile from the population, () is the current best position, () is the random coefficient controlling the intensity of movement, and () is a linear decay coefficient that reduces the exploration rate with increasing iteration. The expression () is also a uniform random value between zero and one used to maintain the diversity of the population. In the exploitation phase, the algorithm focuses on improving positions close to the current best solution. This phase is defined by equation 3:

(3)

where () is a random factor that determines the direction and amplitude of the reptile's movement. The third term in the equation allows the algorithm to avoid getting stuck in local optima by maintaining some randomness. After each update, the values of the positions that are out of the allowed range are controlled by equation 4:

(4)

At each iteration, new positions are evaluated based on the objective function () and the best position is updated until the conditions of Equation 4 are met. The above process continues until the stopping criterion, i.e. the maximum number of iterations (T), is reached. In this research, fitness is defined based on the clustering validity index as Equation 5.

(5)

where the objective function to be maximized and silhouette(...) is the silhouette index of the feature weight vector that the RSA algorithm searches and optimizes. The silhouette index is used as a fitting criterion because it has a more stable performance in data with asymmetric distribution. In the proposed method, the RSA algorithm is used to learn the optimal feature weights and select the distance criterion in hierarchical clustering. Each reptile represents a weight vector w and using it, the distance matrix between the data is calculated in the weighted form of Equation 6:

(6)

where is the th feature value for sample , is the th feature value for sample , and is the squared difference between the th feature value in the two samples. By combining adaptive control between the two search phases and using variable random coefficients, the RSA algorithm is able to strike a good balance between global exploration and local exploitation. This feature has made RSA perform more stable and faster than classical algorithms such as PSO and GA in many data clustering, feature selection, and multidimensional optimization problems. The pseudo-code of the improved cumulative hierarchical clustering using the fruit fly algorithm is given in Table 2.

Table 2. Pseudo-code of the improved cumulative hierarchical clustering

```
Algorithm 1: AHC + RSA
Input: Data matrix X with dimensions (N × d) ← N samples, d features
Step 1: Initialization
        1.   Determine the number of clusters K
        2.   Apply K-Means to reduce the dataset into ( C = {, , ..., } ) centroids, where ().
        3.   Set the parameters of the Reptile Search Algorithm (RSA):
             - Population size ( PopSize )
             - Maximum iterations ( MaxIter )
        4.   Initialize each individual (= [, , ..., ] ) (feature weights) randomly within ([0,1]).
        5.   Normalize each weight vector ().
Step 2: Run RSA
For each iteration ( t = 1 ) to ( MaxIter ):
  1. Compute the adaptive control coefficient:
     ( ).
  2. Set the random weight factor (beta = 0.5 + rand()/2 ).
  3. For each reptile ( i = 1, 2, ..., PopSize ):
     If ( rand < 0.5 ) then    ▷ Exploration phase
       Select a random individual ( ) from the population.
       Update the position using:

     Else    ▷ Exploitation phase
       Set (-1).
```

```
            Update the position using:

      End If
         Apply boundary control:
         ,
         Normalize () → (=).
         Compute the weighted distance matrix for hierarchical clustering:

         Apply Agglomerative Hierarchical Clustering (AHC) using the Ward linkage criterion.
         Partition the data into ( K ) clusters and compute the fitness function:

      End For
      4. Select the best solution (  ).
   End For
   Step 3: Final Clustering
   1. Compute the final normalized weight vector ( ).
   2. Apply AHC on the centroid matrix (C) using the weighted features ( =C ).
   3. Generate the final cluster labels ( L = {L_1, L_2, ..., L_N} ) for all samples.
   4. Compute the final clustering quality metrics (e.g., Davies–Bouldin index, silhouette score).
   Output: Final cluster labels ( L ), Optimal feature weights (), Final fitness value () (e.g., average silhouette
   or DB index)
```

In the time complexity analysis of the combined algorithm of AHC-RSA, two main parts can be examined. Since RSA is a population-based algorithm, a complete hierarchical clustering must be performed for each member of the population in each iteration. Therefore, the overall time complexity of this method is equal to the product of the number of iterations times the population size and the AHC execution time. For a data set with samples and features, the calculation of the distance matrix in AHC requires order , and the hierarchical clustering process will have order at best. In addition, in each evaluation, the silhouette index is used as the objective function, which also depends on the time . As a result, the overall time complexity of the combined algorithm can be expressed as follows:

(7)

where is the size of the fly population, is the number of algorithm iterations, is the number of data samples, and is the number of features. Therefore, as the data volume increases, especially when m is very large, the dominant part of the time complexity will be related to the quadratic calculations in AHC and the evaluation of the silhouette index. For this reason, in large scales, approximate versions or dimensionality reduction and sampling methods are usually used to reduce the computational load and make the algorithm practical. The worst case and time complexity that can be considered for hierarchical clustering is that all users are placed in one cluster, meaning that users who have given specific feedback to cloud service providers will be placed in the same cluster. Or, in front of each user, it is a cluster and we will have as many clusters as users. If these cases occur, according to step 4, the data set analysis with the empirical data analysis method, which is a strong statistical method, will be applied to the entire data set.

3-3- Enhanced-Empirical Data Analysis Method (E-EDA)

In this study, Euclidean distance and density distribution are used to identify anomalies and improve clustering. The input data set where: is the -th data sample, k is the total number of samples, is the set of unique samples, is the frequency of occurrence of each u_i such that and the output is the set of potential anomalies in each cluster. The proposed method consists of four main steps:

Step 1 Global Proximity (Cumulative Neighborhood): For each data sample , the cumulative proximity is defined as the squared Euclidean distance from to all other samples:

(8)

where is the Euclidean distance. Using the mean and the mean square , this measure represents the overall dispersion of data around the mean.

DOI: 10.46121/pspc.54.1.23

Innovation: In E-EDA, is employed not only as a dispersion measure but also as the foundation for local and multimodal density estimation.

Step 2 Unimodal Density: The unimodal density of a data sample is defined as:

(9)

where a factor of 2 is used in the denominator of , because each interval is counted twice in the total density of all data samples. This follows the form of a Cauchy-type function and indicates the degree of concentration of data around the mean.

- Symmetric distribution → balanced tails.
- Skewed right → heavier right tail.
- Skewed left → heavier left tail.

Innovation: Unlike the original EDA, in E-EDA unimodal density is also used to extract distribution asymmetry indices, improving anomaly sensitivity.

Step 3 Multimodal Density: For each unique sample with frequency :

(10)

and for repeated data samples :

(11)

This enables the detection of local structures and hidden clusters without iterative search algorithms.
Innovation: Frequency-weighted multimodal density allows rare outliers (low-frequency points) to be separated more effectively from dense clusters.

Step 4 Chebyshev's Inequality and Local Weighted Density: For non-normal distributions, Chebyshev's inequality is applied:

(12)

Steps:
1. Compute the mean squared Euclidean distance:

(13)

2. Define a local influence region (hyper-sphere) for each with radius .
3. Identify local neighbors and compute the local weighted unimodal density:

(14)

where is the number of unique neighbors and is the number of unique samples.
4. Rank data by ; the lower half (less than) is considered as potential anomalies
Innovation: The integration of local weighted density with Chebyshev's inequality provides a robust mechanism for identifying anomalies in non-normal data distributions. The pseudocode of the experimental data analysis method is given in Table 3.

Table 3. Pseudo-code of the experimental data analysis method

Algorithm 2: Enhanced Empirical Data Analytics (E-EDA)
Input: Clustered samples 1.Start 2. For each cluster , i=1 dots K: a. Compute the local and global multimodal density for each sample: (where is approximated by Mahalanobis distance and by LOF.) b. Identify anomaly candidates: samples with the lowest . 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**

3-4- Extreme Learning Machine Algorithm Optimized with PSO Algorithm

In this research, the ELM algorithm optimized using the PSO algorithm and shown in Figure 4 is used. The ELM algorithm is a technique in which we only need n linear operations to update the weights, where n is the number of input data. An ELM is a type of neural network that has only one hidden layer, and its difference from a conventional neural network is that the values of the weights of the first layer do not change during learning and the only parameters that can be learned are the weights of its output layer. In this algorithm, it is assumed that there are arbitrary samples in the training phase in the form   with number of hidden nodes. The output function of an ELM for a single-layer feedforward neural network with n hidden nodes is represented as . Where is the input weight vector associated with the hidden layer node, is the bias value of the hidden layer nodes, is the output weight vector associated with the hidden layer node, and is the nonlinear feature mapping of the maximal learning machine. The output functions of the nodes may not be unique and may be used in different hidden neural cells. Also , where refers to the output functions of the hidden node (with the parameters of the hidden nodes and is a continuous nonlinear function that satisfies the theorems of the ability to estimate the overall coefficient of the maximum learning machine. In this study, the sigmoid activation function has been used due to its greater use in other research and greater popularity among activation functions. The sigmoid function, also called the logistic function, has an S-shaped shape that can map any value to a value between zero and one. If the curve goes to positive infinity, the prediction is mapped to the number one, and if the curve goes to negative infinity, the prediction is mapped to the number zero. In the PSO algorithm, each particle is considered as a bird, and all birds seek to find the best point, and this happens through information sharing. Information sharing is such that all birds calculate their fitness function, and each bird with the best fitness informs the other birds, causing the other birds to move towards it, and this continues until all birds gather at the best point. Each bird chooses the best direction to move, considering its current location, its own previous best local location, and the best location of other birds. In fact, the assumption of information sharing between birds is the basis of this algorithm. Using the PSO algorithm, the values of the input weights and the hidden layer biases in the ELM algorithm are optimized. is an arbitrary sample group of where and such that: is the input and is the expected output.

Step 1: Data preprocessing. All features of the classification dataset are normalized to the range [1, 0], and then the classification dataset is randomly divided into training and testing data proportionally.

Step 2: P particles are randomly initialized in the range [1, -1] for the bias values of the hidden neurons and [1, -1] for the input weight values, where where z is the population size. The position of the i-th particle is defined as . The velocity of the ith particle is defined as where and the velocity of each particle is . The maximum iteration, denoted by , is given in Table 4.

Table 4. Required variables for maximum iteration with

Algorithm variables	
	Input weight values that indicate the connection between the jth input neuron and the i-th hidden neuron.
	Indicates the bias of the i-th hidden neuron
	Number of input neurons
	Number of hidden neurons
	Indicates the dimensions of the particle whose parameters need to be optimized.
W	Internal weight
,	Acceleration coefficient
,	Random numbers in the range [1 and 0]

Step 3: At this stage, the number of hidden neurons and the activation function of the ELM algorithm can be expressed as equation (15):

(15)

Where is the number of samples, is the number of samples with correct values, is the input weight vector connected to the hidden layer node, is the bias value of the hidden layer nodes, and is the output weight vector connected to the hidden layer node. It is calculated according to equation (16).

(16)

Where and are calculated according to equation (17).

(17)

where is the output matrix of the hidden layer of the extreme machine learning algorithm network. In , column i refers to the i-th hidden layer neuron in the input neurons. is the generalized Moore–Penrose inverse of itself. The activation function is infinitely distinguishable when the desired number of hidden neurons $L \leq N$.
Step 4: In this step, the value of each particle in the population is calculated according to equation (15).
Step 5: In this step, the best position of each particle and also the best position for all particles are calculated. The objective function is then the particle swarm update according to equation (18):

(18)

Step 6: The position and velocity of each particle are recalculated based on equation (19) and (20) based on equation (15).

(19)

. (20)

The constants and in equation 1 are the learning parameters (effect size) for the best particle position. k is the loop counter, W is a parameter that controls the stagnation of the particle motion, and are random numbers in the range [1 and 0].

Step 7: If the stopping criteria are reached, the optimal input weight and the bias of the input hidden layers are stored. This is calculated by optimizing based on the minimum error of the algorithm, otherwise we go to step 4.

Step 8: The results of particle mass optimization are used as the input weight and bias for the ELM algorithm, and the output matrix of the hidden layer (H) is calculated using equation (17).

Step 9: The output weight is calculated based on equation (14) and the prediction model of the ELM algorithm is calculated with the minimum MSE evaluation criteria [39, 40].

The pseudo-code of the ELM algorithm optimized with the PSO algorithm is given in Table 5.

Table 5. Pseudo-code of the optimized ELM algorithm with the PSO algorithm

Algorithm 3: Extreme Learning Machine Algorithm Optimized with Particle Swarm Optimization Algorithm	
Input	dataset
1	Start
2	for i=1 to max_iterations Dataset partitioning and feature normalization Dataset in the range [1, 1-]
3	Generate initial particles and initialize them, position and velocity of each particle
4	Determine the number of hidden neurons and define the activation function of the extreme machine learning algorithm
5	Calculate the value of each particle in the 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 biases of hidden layers input of the extreme machine learning algorithm
9	End
10	Calculation of the output matrix of the hidden layer (H)
11	Calculation of output weight β
12	a. if fitness(i) < elm_fitness(global_best, X_train, Y_train, hidden_neurons)
13	b. global_best = particles(i, :);
14	c. BestMSE = fitness(i);
15	d. End
16	End
Output	Calculation of evaluation criteria

DOI: 10.46121/pspc.54.1.23

EXPERIMENTS AND RESULTS

In this paper, the simulation environment with MATLAB R2020a software is used for modeling, and the computer for simulation has an Intel i3-2350 series central processor with 4 GB memory and 500 GB hard drive.

4-1- Dataset

Two datasets were used in this study. The first dataset: Rich Epinions, which was used in the research of Siadat et al.[2] and Soleymani et al.[44], is licensed by Simon Meyffret, Frédérique Laforest, and Lionel Médini under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The dataset is taken from the site https://projet.liris.cnrs.fr/red, which has been on this site since June 2011. This dataset includes several tables, including a user expertise table for categories, a product category table, a user, a feedback table, a feedback evaluation table, a similarity table, and a trust table between users.

The second dataset, the CloudArmor dataset, used in the research of Truong et al.[7] and Aghaee Ghazvini et al.[8]. Due to the unavailability of the original **CloudArmor dataset** — which was introduced by the original authors years ago and is no longer publicly accessible — a **synthetic simulated dataset** was generated in this study to enable evaluation of the proposed method. The structure of this dataset was designed based on the published descriptions and characteristics of CloudArmor. Specifically, the synthetic dataset includes **user ID, service provider ID, numeric rating** given to the service, **feedback timestamp**, and **review text**, representing the user's qualitative opinion about the service quality (e.g., "Excellent service" or "Poor reliability"). These textual reviews were converted into corresponding numerical values to be compatible with machine learning models. Furthermore, to simulate more realistic conditions, a certain percentage of the feedback was intentionally injected as **fake or deceptive feedback**, allowing assessment of the proposed method's ability to detect such anomalies. The final dataset contains **12,000 feedback entries** from **5,000 users** across **113 services**, providing a controlled yet realistic environment for testing model performance in cloud service scenarios. Out of 12,000 records, 1,800 are considered fake reviews. Fake users typically: Post multiple similar reviews on different services. Use short, exaggeratedly positive sentences. Give very high or very low ratings (1 or 5). Real users have more varied patterns and use more natural sentences.

4-2- Evaluation Measures

In this study, the evaluation criteria used in various studies, including [5, 9, 26, 35, 34, 28, 27], which are given in equations (21) to (24), were used.

(21)
(22)
(23)
(24)

In the above equations, TP and TN are for correct diagnoses and FP and FN are for incorrect diagnoses. MSE and RMSE are also used to calculate the error rate. MSE and RMSE are measures for measuring the error rate in prediction models that calculate the mean square of the differences and its root, respectively, with the difference that RMSE represents the error in the original unit of the data. These measures are obtained according to equations (25) and (26).

(25)
(26)

In the above relationships, is the value predicted by the model and is the actual value.

4-3 Implementation of the proposed method and the Setting hyperparameters

To implement the proposed method, the **Rich Epinions** dataset was first utilized, containing **36,220 records** and **six attributes**, including feedback ID, user ID, rating score, feedback text, product ID, and *timestamp*. Preprocessing operations were performed on the dataset to ensure consistency and quality of the data. The histograms of two key features—user ratings and *feedback toward the cloud service providers*—are illustrated in

Figure 3**, providing an overview of the data distribution prior to model training.**

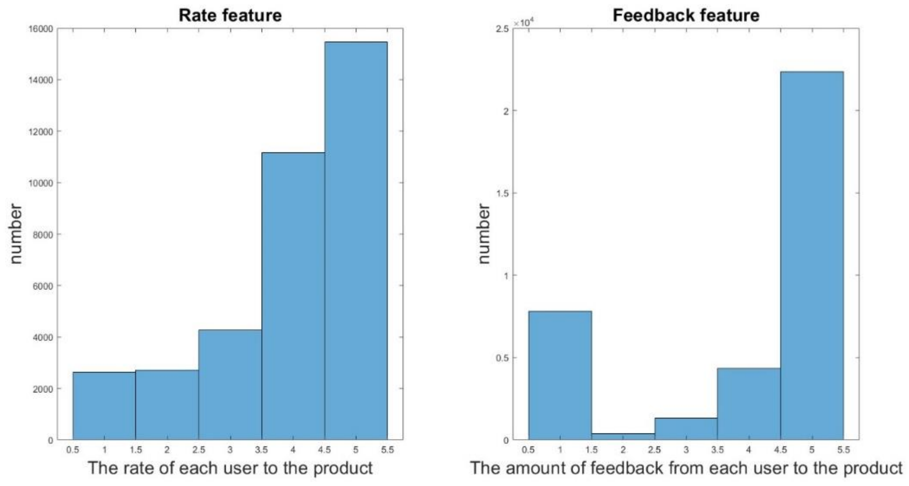

Fig. 3. Histograms of the two features—rating and feedback

Figure 3 illustrates the histograms of the two features—rating and feedback. As observed, the majority of users have provided a rating of 5 and feedback categorized as "Very Helpful." This indicates a positive bias in user evaluations toward service providers.

The next step involves performing Enhanced AHC optimized using the RSA. The parameters utilized for the improved hierarchical clustering process are summarized in Table 6.

Table 6. Parameters of Enhanced AHC optimized with the RSA

Parameter Type	Parameter Name	Symbol / Variable	Value or Range	Description
Input Data	Data Matrix	X	N × d	Contains N samples and d features
Preprocessing Stage	Number of Initial K-Means Cluster	numPreClusters	500	Reduces dataset size from N samples to 500 initial cluster centers
RSA Algorithm	Fly Population Size	PopSize	20	Number of candidate solutions in each iteration
	Maximum Number of Iterations	MaxIter	30	Number of search iterations
	Problem Dimension	dim	d(number of features)	Equal to the number of features in the cluster centers
	Lower Bound of Weights	lb	0	Minimum possible value of each feature weight
	Upper Bound of Weights	ub	1	Maximum possible value of each feature weight
Objective Function	Clustering Evaluation Metric	Fitness	Mean Silhouette	Measures clustering quality based on intra-cluster cohesion and inter-cluster separation
Final Clusters	Number of Clusters	numClusters	2–10 (random or optimized)	Number of clusters generated in each iteration of the algorithm

Figure 4 illustrates the output of the **AHC** performed using the **RSA**. As shown, the optimized hierarchical structure effectively groups similar feedback patterns, demonstrating improved intra-cluster cohesion and inter-cluster separation compared to the conventional AHC approach.

DOI: 10.46121/pspc.54.1.23

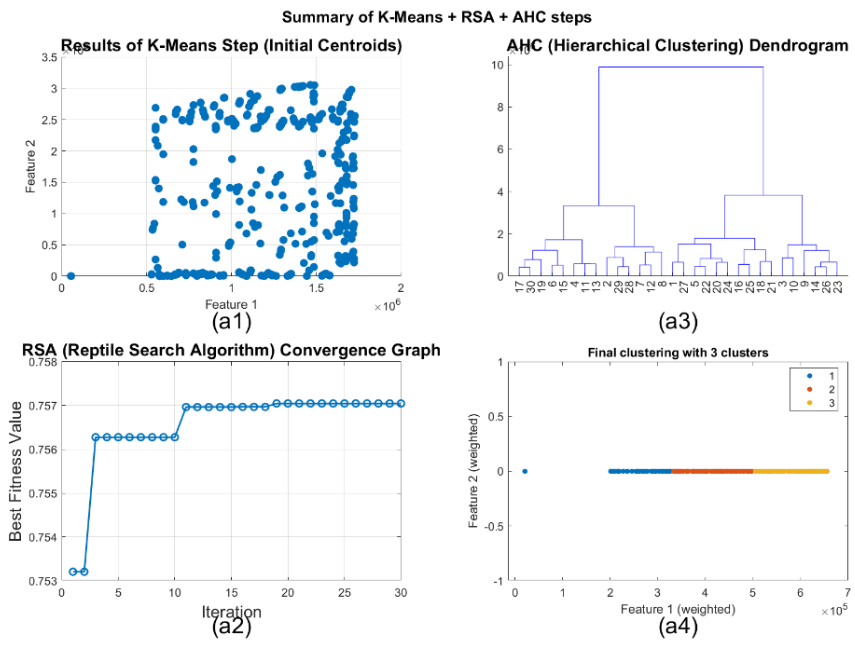

Fig. 4. The clustering results of the dataset obtained using the AHC-RSA

As can be seen in Figure 4, in order to reduce the data volume and prevent high memory consumption in the implementation of the optimization algorithm, in the first step, the K-Means algorithm was used for data preprocessing. In this step, the initial data with N samples and d features were reduced to 500 cluster centers to reduce the input dimensions for the optimization step while maintaining the data distribution structure. These initial centers were used as representatives of the original data in the rest of the process. In the second step, the RSA algorithm was implemented to determine the optimal weights for the features and maximize the silhouette index, because in this study the objective function is defined as maximizing the clustering quality index. The results from the algorithm convergence graph showed that the fit value increased rapidly and reached a stable value close to 0.757 after several iterations, which indicates the effective convergence of the algorithm. In the final step, hierarchical clustering was performed on the weighted data using Ward's linkage method. The resulting dendrogram shows the hierarchical structure of the clusters, and in the end, the data were classified into three separate clusters with appropriate resolution. The results show that the combination of K-Means for data reduction, RSA algorithm for feature weight optimization, and AHC for final clustering improved the stability and quality of clustering compared to the direct use of each method alone. In the final clustering step, the weighted data after optimization by RSA algorithm were displayed in a normalized form. For this reason, the feature values are centered around zero and the distribution of clusters on the horizontal axis of the graph is observed as dense. This is due to the standardization of the data after applying weighting coefficients and is not an indication of an error in the process.

Also, in the clustering process, because users with similar characteristics and views are placed in a cluster, the cluster label can be used as a new feature in the dataset. This feature can reflect the collective behavior of users and hidden decision-making patterns and play an effective role in improving the performance of supervised learning models. In other words, assigning each record to a specific cluster provides a kind of high-level knowledge about the response pattern of users, which, when combined 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 indicates the distribution of data in the three main clusters.

Table 7. Number of samples to each cluster

Cluster	Cluster1	Cluster2	Cluster3
Number of each sample	1934	17709	16577

According to Table 7, the first cluster, with 1934 samples, has the smallest size, while the second and third clusters, containing 17709 and 16577 samples respectively, comprise the major portion of the dataset. This distribution indicates that most data points are concentrated in the second and third clusters, whereas the first cluster represents a smaller, more specific subset of user behaviors.

The next phase of the proposed framework applies the E-EDA method to detect fake feedback within each cluster. This step leverages the cluster-specific characteristics derived from the previous stage to improve anomaly detection accuracy. Table 8 summarizes the number of fake feedback instances identified in each cluster using the E-EDA method.

Table 8. Number of fake feedback instances identified in each cluster using the E-EDA method

Cluster	Cluster1	Cluster2	Cluster3
Number of fake feedback	279	970	1379

According to Table 8, using the improved empirical data analysis method, a total of 2628 records were identified as fake feedback, of which 279 samples belong to the first cluster, 970 samples belong to the second cluster, and 1379 samples belong to the third cluster, as shown in Figure 5.

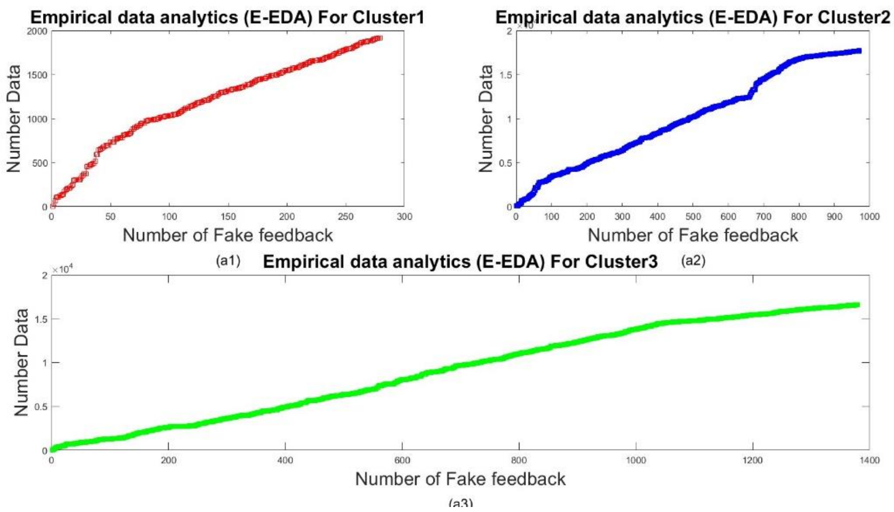

Fig 5. Detection of fake feedback using the E-EDA method

As shown in Figure 5, the highest detection of fake feedback is in the third cluster. The next step is to label the class feature. In the dataset, records that are identified as anomalies and fake feedback are class 1 and the rest of the records are class 0.

In the next stage, the ELM-PSO algorithm was employed for **model training and evaluation**. However, since the number of **normal (genuine)** feedback records significantly exceeds the number of **fake feedback** samples, the dataset suffers from a **class imbalance problem**, which can lead to biased learning. To address this issue, the **Simple SMOTE (Synthetic Minority Over-sampling Technique)** method was applied to balance the dataset. This technique generates **synthetic samples** for the minority class (fake feedback) by averaging feature values and adding random noise, effectively enhancing class distribution without removing real data. The **oversampling rate** — which determines how many synthetic instances are generated — was tuned experimentally to achieve an optimal trade-off between **Precision** and **Recall.** After completing preprocessing and rebalancing, the total number of samples decreased from **36,220** to **27,168**, mainly due to the removal of invalid and redundant records. To ensure fair evaluation and avoid **data leakage**, the dataset was split into **70% training** and **30% testing** sets, and the SMOTE procedure was applied **only** to the training set. **Table 9** summarizes the parameters used for the ELM-PSOalgorithm.

Table 9. **Parameters used for the ELM-PSO algorithm**

Parameter	Value
hidden_neurons	30
particle_count	50
max_iterations	100
=	1.99
=	0.7
input_size	size(datadset, 2)
dimensions	hidden_neurons * (input_size + 1)
	reshape(particle(1:hidden_neurons input_size), [hidden_neurons, input_size])
	particle(hidden_neurons input_size + 1:end)
H	tanh(X * input_weights' + biases')
	pinv(H)*Y

Figure 6 shows the optimized MSE error rate at each iteration and the evaluation results.

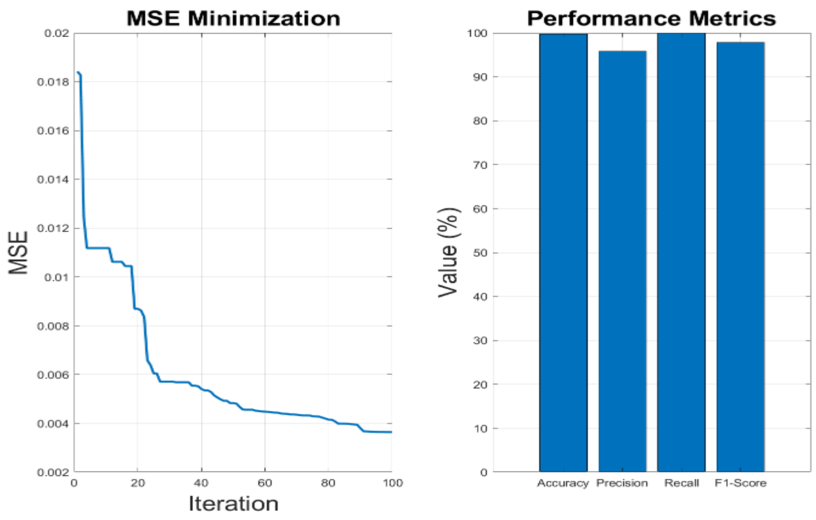

Fig. 6. Optimized MSE error rate per iteration and evaluation results

As shown in Figure 6, the performance of the optimization algorithm is displayed in two figures: “Error Convergence Process” and “Evaluation Results”. The left graph shows the trend of the MSE reduction during 100 iterations. The MSE value decreases from about 0.018 at the beginning of the process to less than 0.003 at about the 80th iteration, and then stabilizes at a constant value. This behavior indicates that the algorithm has successfully achieved the minimum possible error and achieved stable convergence. The right graph shows other evaluation criteria, the full results of which are shown in Table 10. This table shows the output of the optimized extreme machine learning algorithm with the particle swarm optimization algorithm based on the evaluation criteria.

Table 10. Output of the optimized extreme machine learning algorithm with the particle swarm optimization algorithm based on the evaluation criteria

Evaluation Criteria	MSE		Accuracy	Precision	Recall	F1-score
Epinions Dataset	0.0036	0.0604	99.69%	99.82%	100%	97.87%

Based on the evaluation criteria table 10, the proposed model has shown a very accurate and stable performance. The accuracy value of 99.69% and the recall value of 100% indicate that all positive samples have been correctly identified. Also, the prediction Precision of 95.82% and the average F1-score of 97.87% indicate that the model has established a favorable balance between sensitivity and precision. Also, the low values of RMSE = 0.0533 =

0.0604 and MSE = 0.0036 indicate good convergence and very low error of the model in the classification process. These results generally prove that the model has been 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.

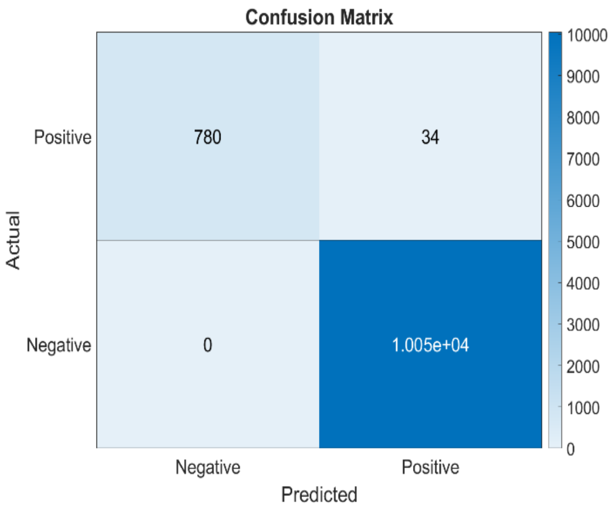

Fig. 7. Confusion Matrix

In Figure 7 and according to Table 10, the confusion matrix of the proposed model is shown. According to the results, the proposed model performed very well on the Epinions dataset. The figure related to the confusion matrix shows the results from 30% of the test data (about 11,000 samples out of a total of 27,168 samples) that the model was able to correctly identify 10050 true negative samples and 780 true positive samples, while only 34 false positives were predicted and no false negative samples were identified. These results indicate the very accurate performance of the model in distinguishing fake and real feedback. The distribution of values in the matrix shows that the model has a precision and recall of close to 100% and has been able to correctly learn the data pattern with minimal classification error. This confirms the high efficiency of the combination of AHC-RSA_E-EDA_ELM-PSO algorithms in identifying fake feedback and improving the quality of prediction in cloud environments. In order to increase the accuracy of the results and prevent bias in the analysis of model performance, the ANOVA statistical test was performed based on the evaluation criteria with 6 independent repetitions. Figure 8 shows the performance of the proposed method based on the ANOVA test.

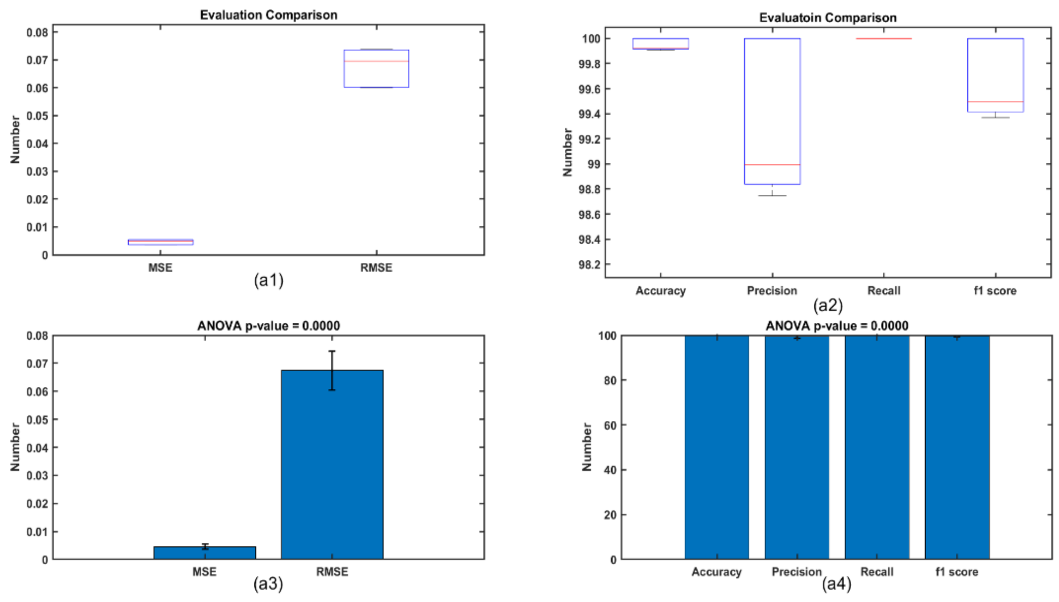

Fig. 8. Performance of the proposed method based on ANOVA test

As can be seen in Figure 8, graphs (a1) and (a3) are related to the comparison of MSE and RMSE errors and show that the values of both indices are very low, equal to 0.0044 and 0.0662, with limited variance, which indicates the stability of the model in prediction error. Also, graphs (a2) and (a4) show the comparison of the accuracy criteria, Precision, recall, and average F1-score. The average accuracy criterion in 10 repetitions is 99.84%, Precision is 98.84%, recall is 100%, and F1-score is 99.36%, all of which are at a level above 98%. The p-value = 0.0000 for both ANOVA tests also indicates the statistical significance of the results and the significant difference between the model performance compared to the compared cases; therefore, the model has high stability and efficiency in several independent runs. Table 11 shows the performance of the proposed method compared with the types of methods used in it.

Table 11. Performance of the proposed method based on the methods used in it

Evaluation Criteria	**MSE**		**Accuracy**	**Precision**	**Recall**	**F1-score**
without clustering	No answer because this set requires 8.9 GB of memory.					
with basic hierarchical clustering (AHC_E-EDA_ELM-PSO)	0.0037	0.0605	79.74%	68.88%	83.78%	75.60%
simple ELM with the enhanced version in (AHC-RSA_E-EDA_ELM)	0.04481	0.2109	96%	62.50%	83.33%	71.43%
with EDA (AHC-RSA_ EDA_ELM-PSO)	0.0079	0.0886	96.91%	96.63%	99.99%	96.28%
Proposed Method (AHC-RSA_E-EDA_ELM-PSO)	0.0036	0.0604	99.69%	99.82%	100%	98.87%

Based on the results obtained from Table 11, the use of hierarchical clustering has reduced memory consumption and relatively improved model performance compared to the case without clustering. In the basic case, running the model without clustering was not possible due to the need for about 8.9 GB of memory. By using the proposed method with simple hierarchical clustering (AHC_E-EDA_ELM-PSO), the F1-score value reached 75.60% and the overall accuracy reached 79.84%. By replacing the simple extreme learning model in the method (AHC-RSA_E-EDA_ELM), the F1-score value reached 71.43% and the overall accuracy reached 96%, which indicates an increase in the convergence and stability of the model in learning the relationships between clusters. Also, the removal of the E-EMD component also led to a relative drop in performance, while the final combination (AHC-RSA_E-EDA_ELM-PSO) recorded the best performance with an F1-score of 97.87%, recall of 100%, and the lowest MSE and RMSE. This analysis shows that the addition of optimization components (RSA) and advanced statistical analysis (E-EDA) directly improved the accuracy and stability of the model in clustering and detecting fake feedback.

For further performance, the proposed method was evaluated on a second dataset, which is related to the cloud environment and is simulated based on the original CloudArmor instance. The only difference is in the discretization of the feedback, which is used for this dataset. The text feedback of the users was mapped to unique numerical values using the ordinal encoding method; In this way, each unique expression in the Feedback_Text variable was replaced with 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 Feedback_Code variable in the fake feedback detection model. Figure 9 shows the AHC-RSA clustering on this dataset.

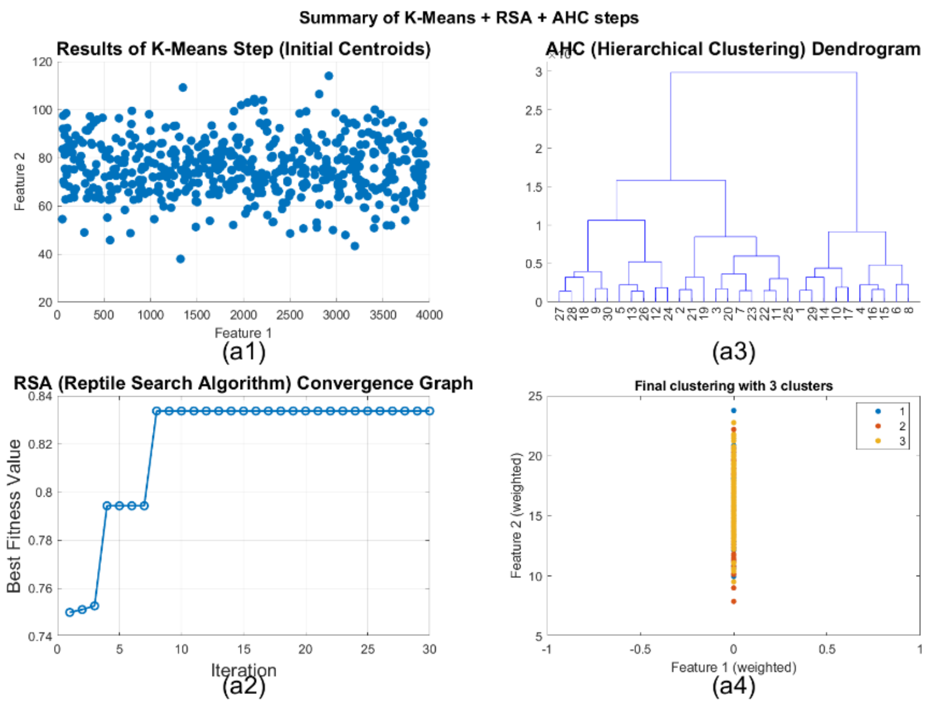

Fig. 9. AHC-RSA clustering on the simulated CloudArmor dataset

Figure 9 shows a summary of the steps of implementing the proposed clustering process based on the combination of K-Means, RSA, and AHC algorithms on the CloudArmor dataset. Figure (a1) shows the results of the initial K-Means stage, (a2) shows the convergence trend of the RSA algorithm in 30 iterations and a stable convergence of about 0.84. This trend shows that the search process effectively converges to the optimal point and the algorithm has high stability and efficiency. Figure (a3) shows the hierarchical clustering dendrogram, and (a4) shows the final clustering result in the form of three distinct clusters. The distribution of data in these three groups indicates high intra-cluster coherence and significant separation between clusters, which indicates the high accuracy of the model in identifying internal patterns in the data. Overall, the results of this figure show that the combination of the three algorithms K-Means, RSA, and AHC has been able to identify and cluster the hidden structure of the CloudArmor cloud environment data with fast convergence and high stability. Figure 10 shows the feedback loop with E-EDA on the simulated CloudArmor dataset.

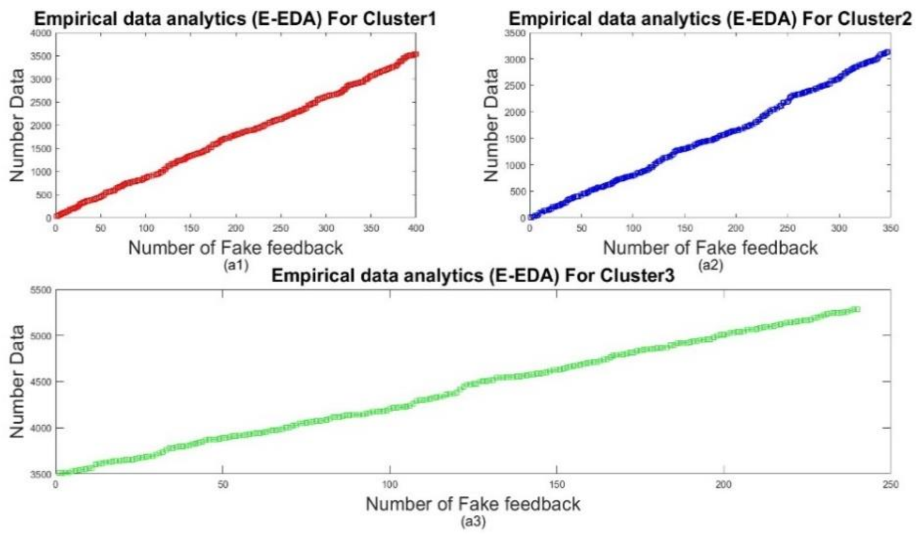

Fig. 10. Detection of fake feedback on the simulated CloudArmor dataset

As Figure 10 shows, from the improved experimental data analysis method, a total of 987 records were identified as fake feedback, of which 400 samples belonged to the first cluster, 347 samples belonged to the second cluster, and 240 samples belonged to the third cluster. This method was able to detect more than half of the feedback. Figure 11 shows the trend of MSE reduction and the results of the benchmarks in this dataset.

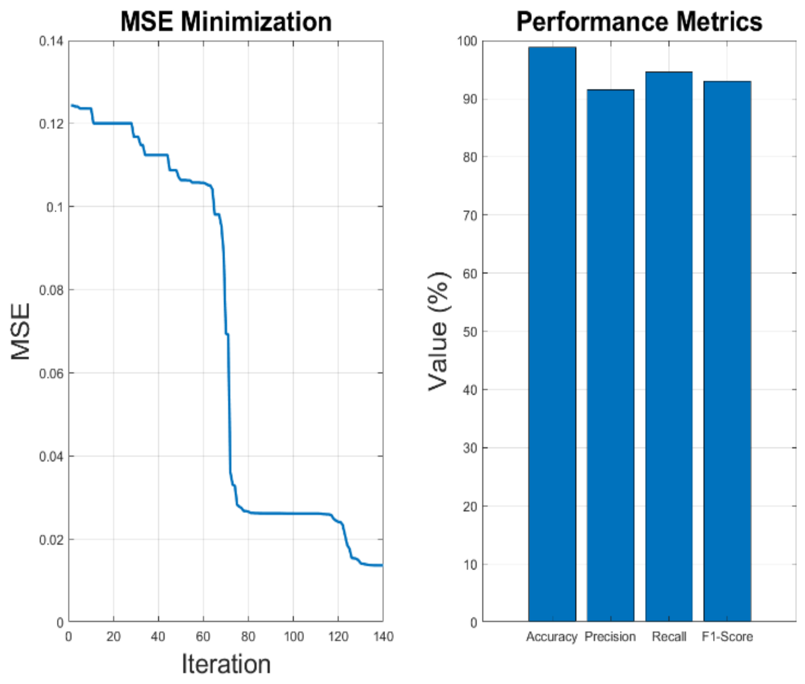

Fig. 11. shows the MSE reduction trend and the results of the criteria in this simulated CloudArmor dataset

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

Table 12. Output of the optimized extreme learning machine algorithm with PSO optimization algorithm based on the evaluation criteria for the simulated CloudArmor dataset

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

According to Figure 11 and Table 12, in the simulated CloudArmor dataset, the proposed model has achieved an overall accuracy of 98.83%, which indicates its high power in correctly identifying the samples. Also, the F1-score value of 93.02% and recall value of 94.59% indicate a good balance between accuracy and recall of the model. The low RMSE and MSE values also indicate good convergence and low model error in the prediction process. Overall, these results confirm the reliable performance and stability of the model in detecting fake feedback in the cloud environment. Figure 12 shows the confusion matrix of the results obtained by the proposed method on the simulated CloudArmor dataset.

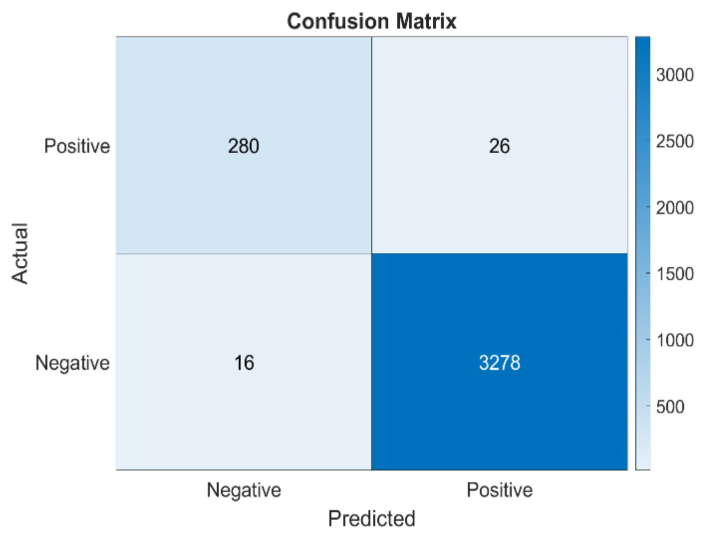

Fig. 12. Confusion matrix of simulated CloudArmor dataset

In Figure 12 and according to Table 13, the confusion matrix of the proposed model has performed very well on the simulated CloudArmor dataset. The figure for the confusion matrix shows the results from 30% of the test data (about 3600 samples out of a total of 12000 samples) that the model was able to correctly identify 3278 true negative samples and 280 true positive samples, while only 26 false positives were predicted and 16 false positives were not identified. These results indicate the performance of the model in distinguishing fake and real feedback. The distribution of values in the matrix shows that the model has an accuracy of 98.83% and has been able to correctly learn the data pattern with minimal classification error. This confirms the high efficiency in detecting fake feedback and improving the quality of prediction in cloud environments.

To demonstrate the performance of the proposed method with ELM optimized with PSO, the dataset was prepared with the same clustering procedure as AHC-RSA, E-EDA and labeling and other machine learning algorithms were implemented and the results are given in Table 13.

Table 13. Comparison of the performance of the proposed method with other machine learning algorithms

Machine Learning Algorithms	Evaluation Criteria	Epinions Dataset	Simulated CloudArmor Dataset
Decision Tree(NumberTree=100)	MSE	0.032199	0.04111
		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
		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
		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
		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 Neuron=100)	MSE	0.13425	0.021111
		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 Neuron=40)	MSE	0.0075235	0.019722
		0.086738	0.14044
	Accuracy	99.24%	98.02%
	Precision	100%	80%
	Recall	89.67%	100%
	F1-score	94.55%	88.88%
Proposed Model(ELM-PSO)	MSE	0.0036	0.0137
		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 Table 13 shows, different machine learning algorithms showed different performance on the two datasets Epinions and CloudArmor. Classical algorithms such as Decision Tree and KNN had relatively high accuracy at the level of about 96%, but performed poorly in detecting fake feedback due to lower recall (55.32% and 89.06%, respectively) on the Epinions dataset. In contrast, SVM with 98% accuracy and 93.75% recall provided a balanced performance but had a larger fluctuation in MSE error, indicating its sensitivity to data changes. Among the neural networks, ANN had high accuracy but its recall was lower than other models, indicating its tendency to overfit. LSTM and GRU algorithms performed better than the previous models due to their ability to learn temporal dependencies and maintained an accuracy above 98%. Overall, the proposed algorithm (ELM-PSO) has been able to outperform other models by maintaining high stability and the lowest error rate, especially in reducing MSE and RMSE errors and achieving the desired balance between accuracy and recall.

4-5- Comparison the proposed method with other methods

Because the use of machine learning algorithms in cloud computing and this field is very low and the studies have used different data sets, the comparison of the results obtained with other studies is given in Table 14. Two studies, Siadat et al.[2] and Soleymani et al.[44], used the Epinions data set, but in both works, machine learning algorithms and its evaluation criteria were not used. In the paper by Siadat et al.[2], a Bayesian game model was used to investigate and identify malicious users and prevent them from sending feedback. Simulations show that this model can correctly identify malicious users and identify fake feedback. In this paper, the work has been simulated and the results obtained based on false negative and false positive have been used, with the value of false negative being 0.60 and false positive being 0.2, which shows that the difference between the trust distribution before and after the injection of fake feedback was 60%. After using the Bayesian game model and identifying fake feedback, this difference has been reduced to 5%. This indicates a 55% improvement in the accuracy and precision of the trust distribution. Accordingly, in the proposed model of this paper, the accuracy has been significantly improved. In the paper of Soleymani et al. [44], which has worked on the trust model with fuzzy rules, it has been shown that the average level of confidence and trust is reduced by about 29%. This indicates 71% in the accuracy and precision of the diagnosis. However, because in these two papers, the evaluation criteria that are not explicitly stated in this research cannot be included in the comparison table. However, based on the two criteria given in the papers of Siadat et al. [2] and Soleymani et al. [44], the proposed method based on the perturbation matrix of this paper has performed better.

Table 14. Comparison of the proposed method with other studies

	Dataset	**Model**	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
Deshai & Rao[27]	Amazon	CNN-PSO	99.4	98.53	98.01	99.02
Taneja & Kaur[5]	CloudArmor	Ensemble model	97.51	98.19	93.65	95.86
Alsubari et al.[9]	Trip Advisor	Random forest (RF)	95	-	-	-
Martens & Maalej [35]	App Apple	MLP	91			
Javed et al.[34]	Yelp	ensemble deep learning	-	-	-	92
thirunavuk karasu et al. [28]	Yelp	Logistic Regression	88.43	-	-	-
Choi et al.[29]	selenium	SVC-Kmeans	85	-	-	-
Proposed Model	Epinions Dataset	AHC-RSA_E-EDA_ ELM-PSO	99.84%	98.84%	100%	99.36%
Proposed Model	Simulated **CloudArmor Dataset**	AHC-RSA_E-EDA_ ELM-PSO	98.83%	91.50%	94.59%	93.02%

Based on the results in Table 14, the comparison between the proposed model and other previous studies shows that the combined AHC-RSA_E-EDA_ELM-PSO model has a superior and more stable performance than the conventional methods in identifying fake reviews. While reference models such as CNN-PSO have achieved 99.4% accuracy and 99.02% F1-score on the Amazon dataset, and combined models such as Ensemble model have performed well on the CloudArmor dataset with 97.51% accuracy, the proposed model has achieved the highest precision and recall with 99.84% accuracy and 99.36% F1-score on the Epinions dataset. In addition, the proposed model also performed remarkably well on the simulated CloudArmor dataset with an accuracy of 99.83%, demonstrating its high generalizability across different data environments. In comparison, more classical models such as Logistic Regression and SVC-Kmeans achieved accuracies of only 88.43% and 85%, respectively, indicating their weakness in extracting complex patterns. Overall, the results indicate that the combination of hierarchical clustering, RSA optimization, and particle swarm optimization-optimized extreme learning significantly improves the accuracy, recall, and robustness of the proposed model compared to other methods. The results also show that the proposed model (AHC-RSA_E-EDA_ELM-PSO) has significantly improved in accuracy and stability compared to all previous methods. Also, in the Epinions dataset, the F1-score and accuracy of the proposed model have increased compared to the CNN-PSO model, indicating an improvement of about 0.8% in overall accuracy and 0.58% in F1-score. In the CloudArmor dataset, the performance of the proposed model has also improved compared to the best existing method, the Ensemble model, in terms of accuracy. These results show that the combination of hierarchical clustering algorithms, RSA optimization, and extreme learning optimized with PSO has increased the detection accuracy and reduced the classification error compared to previous models.

CONCLUSION

Cloud computing, as one of the key technologies of modern computing, provides a dynamic, scalable, and flexible platform for delivering services. However, the growing volume of user feedback in cloud systems has created challenges such as fake feedback, unstable trust, and a decline in service quality. To address these challenges, this study proposed a hybrid framework based on statistical analysis and machine learning, combining improved hierarchical clustering with the crawling search algorithm (AHC-RSA), enhanced empirical data analysis (E-

EDA), and anextreme learning model optimized by particle swarm optimization (ELM-PSO). This integrated design aimed to enhance accuracy, stability, and convergence speed in fake feedback detection.

The main innovations of the proposed research are summarized in three dimensions:
1. Feature weight optimization using the crawling search algorithm, which significantly reduced intra-cluster dispersion and improved inter-cluster separability;
2. The use of the E-EDA statistical model to filter out anomalies and enhance input data quality, leading to more robust learning performance;
3. Integration of ELM-PSO, which accelerated learning, minimized classification errors, and improved prediction accuracy.

Experimental evaluation on Epinions and CloudArmor datasets demonstrated that the proposed approach outperformed traditional algorithms such as SVM, ANN, LSTM, and GRU in terms of accuracy, recall, and robustness. Notably, the proposed hybrid model achieved 99.84% accuracy with 99.36% F1-score on the Epinions dataset and 98.83% accuracy with 93.02% F1-score on the simulated CloudArmor dataset. These results represent an average improvement of approximately 5–8% in accuracy and F1-score compared to the best-performing existing models.

A comparative analysis with recent studies further highlights this improvement. For instance, CNN-PSO achieved 99.4% accuracy on Amazon data [27], Ensemble models obtained 97.51% accuracy on CloudArmor [5], while traditional classifiers such as Random Forest** and Logistic Regression achieved around 91–95% accuracy [9,28]. However, the AHC-RSA_E-EDA_ELM-PSO model surpassed these results, offering superior accuracy, recall, and stability across both datasets.

Furthermore, by combining hierarchical clustering and statistical analysis, the proposed framework enhanced data quality, reduced memory consumption, and increased noise resistance, especially in unbalanced and large-scale datasets. The integration of E-EDA and ELM-PSO also improved model convergence speed and error resilience, as reflected in the lowest MSE and RMSE values among all compared methods.

In conclusion, the proposed framework demonstrates high potential for real-world deployment in cloud-based reputation and feedback management systems. Its ability to achieve a robust balance between accuracy, recall, and generalization makes it a reliable and scalable solution for detecting fake feedback.

Challenges and Future Work: Although the proposed model showed impressive performance, challenges such as the sensitivity of FFOA and PSO algorithms to data size, the dependence of the model on balanced data, the lack of testing in real-time environments, and the possibility of class overlap in the SMOTE method still exist. In addition, the generalizability of the model to other cloud domains requires further investigation.

Future work can extend this research by exploring adaptive parameter tuning, deep hybrid optimization, and real-time implementation to further enhance scalability and performance in dynamic cloud environments.

REFERENCES

1. Islam, R. Patamsetti, V. Gadhi, A. Madhavi Gondu, R. Bandaru, Ch. M. Kesani, S. Ch. Abiona, O. (2023). The Future of Cloud Computing: Benefits and Challenges. International Journal of Communications, Network and System Sciences. Vol, 16. No, 4. Pp: 53-65. DOI: 10.4236/ijcns.2023.164004.
2. Siadat, S. Rahmani, A. M. Navid, H. (2017). Identifying fake feedback in cloud trust management systems using feedback evaluation component and Bayesian game model. Springer, The Journal of Supercomputing. Vol, 73. No, 6. pp: 2682–2704. doi.org/10.1007/s11227-016-1950-1.
3. Mujawar, Tabassum N. Bhajantri, Lokesh B. (2022). Behavior and feedback based trust computation in cloud environment. Elsevier, Journal of King Saud University - Computer and Information Sciences. Vol, 34. No, 8. Pp: 4956-4967. doi:10.1016/j.jksuci.2020.12.003.

4. Guo, R. Tafti, A. Subramanyam, R. (2023). Internal IT modularity, firm size, and adoption of cloud computing. Springer, Electron Commer Res. Pp: 1-30. https://doi.org/10.1007/s10660-023-09691-8.
5. Taneja, H. Kaur, S. (2021). An ensemble classification model for fake feedback detection using proposed labeled CloudArmor dataset. Elsevier, Computers & Electrical Engineering. Vol, 93. No, 1. Pp: 1-15. doi:10.1016/j.compeleceng.2021.10721710.1016/j.compeleceng.2021.107217.
6. Ennaouri, M. Ahmed, Z. (2022). Fake Reviews Detection through Machine learning Algorithms: A Systematic Literature Review. Springer Nature. Pp: 1-23. DOI:10.21203/rs.3.rs-2039197/v1.
7. Truong, X. Nguyen, Th. T. Tran, V. Kh. 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. Anzmac, Australasian Marketing Journal, Vol. 32. no, 1. pp: 76–90. doi.org/10.1177/144135822211460.
8. 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. Springer, The Journal of Supercomputing. Pp: 1-31. DOI 10.1007/s11227-020-03348-1.
9. Alsubari, S. N. Deshmukh, S. N. Alqarni, Ah. A. Alsharif, N. Theyazn Aldhyani, H. H. Alsaade, Fawaz Waselallah. Khalaf, Osamah I. (2022). Data Analytics for the Identification of Fake Reviews Using Supervised Learning. Computers, Materials & Continua. Vol, 70. No, 2. Pp: 3189-3204. https://doi.org/10.32604/cmc.2022.019625.
10. Robin, T. I. (2019). Cloud based Framework for Fake Review Detection. Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence. Vol, 19. No, 4. Pp: 9-12. DOI:10.34257/GJCSTDVOL19IS4PG9.
11. Udaykumar, S. Latha, T. (2017). Trusted computing model with attestation to assure security for software services in a cloud environment. Int. J. Intell. Eng. Syst. Vol, 10. No, 1. Pp: 144–153.
12. Jagpreet, S. Sarbjeet, S. (2017). Improved topsis method based trust evaluation framework for determining trustworthiness of cloud service providers. J. Grid Comput. Vol, 15. Pp: 81–105.
13. Shilpa, D. Rajesh, I. (2018). Evidence based trust estimation model for cloud computing services. Int. J. Network Security. Vol, 20. No, 2. Pp: 291–303.
14. Liu, Y. Pang, B. (2018). A unified framework for detecting author spa-micity by modeling review deviation. Expert Systems with Applications. Vol, 112. Pp: 148–155.
15. Saumya, S. Singh, J. P. (2018). Detection of spam reviews: A sentiment analysis approach. CSI Transactions on ICT. Vol, 6. No, 2. Pp: 137–148.
16. Ye, J. Akoglu, L. (2015). Discovering opinion spammer groups by network footprints [Conference session]. Joint European conference on machine learning and knowledge discovery in databases. Springer. Pp: 267–282.
17. Manaskasemsak, B. Tantisuwankul, J. Rungsawang, A. (2023). Fake review and reviewer detection through behavioral graph partitioning integrating deep neural network. Neural Computing and Applications. Vol, 35. Pp: 1169–1182.

18. 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. Vol, 34. No, 6. Pp:1621–1641.
19. 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. Vol, 193. Pp: 105520.
20. 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. Vol, 149. Pp: 884–900.
21. 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. Vol, 157. Pp: 113765.
22. Goswami, K. Park, Y. Song, C. (2017). Impact of reviewer social interaction on online consumer review fraud detection. Journal of Big Data. Vol, 4. No, 1. Pp. 1–19.
23. Ahmed, H. Traore, I. Saad, S. (2018). Detecting opinion spams and fake news using text classification. Security and Privacy. Vol, 1. No, 1. Pp. 1–15.
24. Barbado, R. Araque, O. Iglesias, C. A. (2019). A framework for fake review detection in online consumer electronics retailers. Information Processing & Management. Vol, 56. No. 4. Pp. 1234–1244.
25. 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 , Vol, 32. No, 23. Pp: 17259–17274.
26. Soleymani, M. Abapour, N. Taghizadeh, E. Siadat, S. Karkehabadi, R. (2021). Fuzzy Rule-Based Trust Management Model for the Security of Cloud Computing. Hindawi Mathematical Problems in Engineering. Pp: 1-14. doi.org/10.1155/2021/6629449.
27. Deshai. N. Rao, Bhaskara B. (2023). Unmasking deception: a CNN and adaptive PSO approach to detecting fake online reviews. Springer, Soft Computing. Pp: 1-22. doi: 10.1007/s00500-023-08507-z.
28. Thirunavukkarasu, M. Kavya, P. Mahalakshmi, P.T. (2023). Fake Reviews Detection using Supervised Machine Learning. International Journal of Engineering Research and Applications, Vol, 13. No, 4. Pp: 250-254. DOI: 10.9790/9622-1304250254.
29. Cho, W. Nam, K. Park, M. Yang, S. Hwang, S. Oh, H. (2023). Fake review identification and utility evaluation model using machine learning. Machine Learning and Artificial Intelligence. Vol, 5. Pp: 1-13. doi.org/10.3389/frai.2022.1064371.
30. Asaad, W. H. Allami, R. Yossra, H. A. (2023). Fake Review Detection Using Machine Learning. Revue d'Intelligence Artificielle. Vol, 37. No, 5. Pp. 1159-1166. doi.org/10.18280/ria.370507.
31. Ram, N. Ch. S. Vakati, G. Nadimpalli, J. V . Sah, Y. Datla, S. K. (2022). Fake Reviews Detection Using Supervised Machine Learning. Journal Statistics & Approval Details. Pp: 1-10. doi.org/10.22214/ijraset.2022.43202.
32. Salminen, J. Kandpal, Ch. Kamel, A. M. Jung, S. Jansen, B. J. (2022). Creating and detecting fake reviews of online products. Elsevier, Journal of Retailing and Consumer Services. Vol, 64. Pp: 1-15. doi.org/10.1016/j.jretconser.2021.102771.

33. He, Sh. Ollenbeck, B. Overgoor, G. Proserpio, D. Osyali, A. (2022). Detecting fake-review buyers using network structure: Directevidence from Amazon. PNAS. Vol, 119. No, 47. Pp: 1-5. doi.org/10.1073/pnas.221193211.
34. Javed, M. S. Majeed, H. Mujtaba, H. Beg, Mirza O. (2021). Fake reviews classification using deep learning ensemble of shallow convolutions. Springer, Journal of Computational Social Science. Vol, 4. No, 1. Pp: 883–902. doi.org/10.1007/s42001-021-00114-y.
35. Martens, D. Maalej, W. (2019). Towards understanding and detecting fake reviews in app stores. Springer, Empirical Software Engineering (2019). Vol, 24. Pp: 3316–3355. doi.org/10.1007/s10664-019-09706-9.
36. 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. Vol, 87. Pp: 602-608.
37. Shi, P. Zhao, Zh. Zhong, H. Shen, H. Ding, L. (2020). An improved agglomerative hierarchical clustering anomaly detection method for scientific data. Concurrency and Computation: Practice and Experience. doi:10.1002/cpe.6077.
38. Gu, X. (2017). Autonomous Anomaly Detection. 2017 Evolving and Adaptive Intelligent Systems (EAIS). Pp: 1-8. DOI: 10.1109/EAIS.2017.7954831.
39. Albadr, M. A. A. Tiun, S. Ayob, M. AL-Dhief, F. T. (2024). Particle Swarm Optimization-Based Extreme Learning Machine for COVID-19 Detection. Springer, Cogn Comput, Vol, 16. Pp: 1858–1873. https://doi.org/10.1007/s12559-022-10063-x.
40. Yu, L. Danninga, Z. Hongbinga, C. (2015). Prediction of length-of-day using extreme learning machine. Elsevier, Geodesy and Geodynamics. Vol, 16. No, 2. Pp:151–159. http://dx.doi.org/10.1016/j.geog.2014.12.007.