scientific reports

Check for updates

OPEN

Comparative evaluation of natural language processing approaches with particle swarm optimized LightGBM for anomaly detection in cloud system logs

Slavisa Djukanovic[1,8], Sasa Stojanovic[1,8], Bosko Nikolic[1,8], Snezana Anetic[2,8], Tamara Zivkovic[2,8], Miodrag Zivkovic[2,8], Vladimir Simic[3,4,5,8] & Nebojsa Bacanin[2,6,7,8]✉

Effective log assessment is a cornerstone for ensuring the reliability and smooth functioning of large-scale, cloud-based infrastructures. As these environments continuously grow in scope and intricacy, traditional monitoring approaches often struggle to keep up, leading to unnecessary computational strain and diminished system performance. To address these challenges, this research introduces a hybrid framework that integrates natural language processing (NLP) with an optimized LightGBM classifier, tailored for anomaly detection in cloud-generated system logs. Central to the design is a thorough preprocessing pipeline that filters irrelevant information and minimizes bias, thereby sharpening anomaly detection accuracy. The experimental setup explored multiple NLP-based preprocessing strategies, including TF-IDF, BERT, and Word2Vec implementations (employing spaCy and Gensim). Classification relied on the LightGBM model, whose hyperparameters were refined using a customized particle swarm optimization (PSO) metaheuristic. This modified optimization routine boosted both predictive accuracy and model robustness. Results from the study reveal that the combined system significantly enhanced both the identification and classification of anomalies in cloud logs. The most effective configuration achieved up to 100% accuracy under the evaluated experimental conditions, highlighting the framework's potential to strengthen security within cloud ecosystems. To ensure statistical soundness, extensive comparative evaluations were conducted across all configurations. Additionally, model interpretability was improved through SHapley Additive exPlanations (SHAP), which provided transparent insights into the role of individual features in shaping classification outcomes.

Keywords Cloud system logs assessment, Anomaly identification, Cybersecurity, LightGBM, Stochastic optimization, Particle swarm optimization, Natural language processing, Swarm intelligence, SHAP

System logs are chronological sequences of records that document various dimensions of system behavior and operational events. Each log entry generally contains several core elements: a timestamp pinpointing when the event occurred, a severity indicator reflecting its relative importance, the process or component that generated it, and a descriptive message summarizing the occurrence. The timestamp situates the event within a timeline, the severity level distinguishes urgent incidents from routine activity, the source attribute traces the origin to a subsystem, and the accompanying message provides the narrative context. With the rapid growth in scale,

[1]School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, Belgrade 11000, Serbia. [2]Faculty of Informatics and Computing, Singidunum University, Danijelova 32, Belgrade 11000, Serbia. [3]University of Belgrade, Faculty of Transport and Traffic Engineering, Vojvode Stepe 305, Belgrade 11010, Serbia. [4]BEU-Scientific Research Center, Baku Engineering University, Khirdalan City, Hasan Aliyev str. 120, AZ0101 Baku, Azerbaijan. [5] Faculty of Engineering, Dogus University, 34775 Istanbul, Türkiye. [6]Saveetha School of Engineering, SIMATS, Thandalam, Chennai 602105, India. [7]Department of Pathological Physiology, Sechenov First Moscow State Medical University (Sechenov University), 119991, Moscow, Russia. [8]Slavisa Djukanovic, Sasa Stojanovic, Bosko Nikolic, Snezana Anetic, Tamara Zivkovic, Miodrag Zivkovic, Vladimir Simic and Nebojsa Bacanin contributed equally to this work. ✉email: nbacanin@singidunum.ac.rs

distribution, and complexity of modern computing infrastructures, reliance on manual inspection or rigid rule-based monitoring has become increasingly impractical. These traditional methods are hampered by delays, inefficiencies, and susceptibility to oversight, ultimately failing to deliver the real-time visibility required for managing large-scale cloud systems[1,2].

Log interpretation constitutes a critical phase within the analysis workflow, where raw, unorganized textual records are converted into structured, computationally interpretable formats. Despite the existence of numerous parsing methodologies, many remain reliant on predefined heuristics and manual setup. Consequently, their precision and flexibility tend to diminish when confronted with the inconsistency and diversity inherent in practical logging data from large cloud systems[3]. To overcome such shortcomings, machine learning (ML) has gained traction as a compelling substitute, delivering automated mechanisms for uncovering abnormal patterns in system logs. In recent years, a wide spectrum of ML-driven approaches for anomaly identification has been introduced[4].

The utilization of ML for detecting anomalies in cloud-generated logs has grown rapidly, largely due to its capacity to uncover complex hidden patterns found within enormous datasets and to adjust dynamically to novel or unforeseen system failures[2]. Despite this progress, real-world integration of ML in these contexts continues to face substantial hurdles. Principal difficulties include the limited availability of large-scale, well-annotated datasets and the heavy preprocessing demands of raw, text-dense log streams[1]. Furthermore, attaining robust predictive outcomes hinges not only on selecting an appropriate model architecture but also on finely tuning hyperparameters–an optimization challenge broadly recognized as NP-hard[5]. To counteract the obstacles presented by unstructured log text, natural language processing (NLP) approaches are frequently applied to transform logs into structured vectorized forms suitable for ML workflows. Commonly adopted techniques encompass term frequency–inverse document frequency (TF-IDF) and BERT[5]. For example, BERT's contextual embeddings capture nuanced semantic information within log entries, substantially improving the precision of downstream anomaly detection tasks.

Despite the growing body of work on anomaly detection in system logs, several challenges remain insufficiently addressed. In particular, there is a lack of systematic studies that jointly evaluate the impact of diverse NLP representations and advanced hyperparameter optimization strategies on model performance in cloud log analysis. Existing approaches often focus either on feature extraction techniques or on classifier optimization in isolation, without thoroughly investigating their combined effect.

Moreover, many studies rely on fixed or heuristically chosen model configurations, which may lead to suboptimal performance and limit adaptability across different log representations. This creates a gap in understanding how optimization-driven model tuning interacts with varying text embeddings in large-scale, real-world log data. To address this gap, the present study proposes a unified framework that integrates multiple NLP-based preprocessing techniques with a metaheuristically optimized LightGBM classifier, enabling a systematic comparative evaluation of their combined impact on anomaly detection performance.

Cloud-based log anomaly detection presents several interconnected challenges, including the unstructured and high-dimensional nature of log data, the variability introduced by different text representations, and the difficulty of selecting optimal model configurations in complex search spaces. These challenges are further compounded in large-scale environments, where both computational efficiency and detection reliability are critical.

To address these issues, the present study adopts a design that integrates NLP techniques for effective feature representation, a gradient boosting model for scalable and efficient classification, and a metaheuristic optimization strategy to improve hyperparameter selection. In this context, the proposed framework is not constructed arbitrarily, but is motivated by the need to jointly address representation quality, model expressiveness, and optimization complexity within a unified approach.

However, exploring advanced directions in ML invariably exposes researchers to a range of obstacles, among which hyperparameter optimization emerges as particularly complex. Identifying an optimal set of hyperparameters is essential for achieving peak predictive capability, yet the process is infamously intricate and resource-intensive[6]. Moreover, the task is commonly classified as NP-hard, underscoring both the exponential growth of its search space and the lack of universally efficient algorithms to resolve it. To address this issue, metaheuristic optimization methods have risen in importance[7], as they excel at investigating enormous, multidimensional search domains, offering robust and adaptable strategies for determining effective parameter configurations in the context of model tuning.

This study was driven by the pressing demand to develop and rigorously assess strategies for anomaly detection in cloud systems log data, with the overarching aim of strengthening the security and dependability of cloud infrastructures. The experimental architecture integrated NLP with ML classifiers. Multiple text preprocessing techniques were explored, including TF-IDF[8], BERT[9], Word2Vec[10], spaCy, Gensim[11] pretrained on the Google News corpus (default vector dimension 300), and a custom-trained Gensim model (vector dimension 100). Each representation was evaluated in conjunction with the LightGBM classifier[12] to measure its effectiveness in detecting anomalies across large-scale log datasets. To support real-world applicability, lightweight configurations of LightGBM were further tested for deployment on ESP32 boards and other resource-limited microcontrollers. Recognizing the pivotal role of hyperparameter tuning in enhancing predictive capacity, the framework incorporated a tailored version of the particle swarm optimization (PSO) algorithm[13]. This addition enabled systematic fine-tuning of LightGBM's parameters, ultimately boosting both the stability and accuracy of the model.

The proposed framework was empirically assessed using an openly available dataset comprising authentic cloud log records. To guarantee methodological rigor, the evaluation incorporated extensive statistical testing of the outcomes. Additionally, the highest-performing models were examined through SHapley Additive exPlanations (SHAP)[14], a procedure that enhanced transparency by highlighting the relative influence of

individual features and elucidating the rationale underlying specific classification outputs. Taken together, the contributions of this research extend both conceptual insight and practical applicability, particularly across the following dimensions:

- Addressing the lack of integrated analysis between text representation methods and optimization-driven model tuning in cloud log anomaly detection;
- A systematic comparative evaluation of multiple NLP-based representations (TF-IDF, BERT, spaCy, and Word2Vec variants) within a unified and optimization-aware framework;
- The development of an enhanced PSO approach for improving hyperparameter tuning stability and efficiency in LightGBM models;
- A comprehensive performance assessment across multiple evaluation metrics and repeated runs, highlighting the interaction between feature representation, model configuration, and optimization;
- The integration of SHAP-based interpretability to provide insight into model behavior and support practical decision-making in anomaly detection.

The rest of the manuscript is structured as follows. "Related works" reviews existing literature, highlighting prior contributions, unresolved challenges, and the technological foundations that motivate this work. "Methods" describes the optimization strategy employed and elaborates on the core elements of the proposed architecture. "Simulation setup" outlines the experimental design and configuration used for system assessment. "Simulation results" reports the results, while "Validation and interpretation" interprets the outcomes of the evaluation. Lastly, "namerefconclusion" summarizes the principal findings, acknowledges current limitations, and suggests avenues for future research.

Related works

In recent years, research and innovation in text analysis and natural language understanding have expanded rapidly, largely propelled by the advent of advanced transformer-based architectures such as generative pre-trained transformers (GPT)[15] and BERT[9]. These systems, grounded in the transformer paradigm, showcase remarkable proficiency in processing vast textual datasets, generating coherent summaries, and discerning subtle layers of meaning within language. Work published in 2024[16] highlighted the pronounced disparities in both effectiveness and computational efficiency that emerge when these models are applied across a spectrum of linguistic and text-centric tasks.

Designing a robust ML pipeline demands considerable expertise, extensive computational capacity, and careful planning. Beyond solely choosing a suitable model, one of the most critical steps involves fine-tuning its hyperparameters to achieve an effective and well-balanced configuration[17]. This stage, known as hyperparameter optimization, poses a particularly formidable challenge in practical AI applications. The task is NP-hard, owing to the vastness and intricacy of the search spaces as well as the heavy reliance of outcomes on the chosen optimization strategy. According to the no free lunch (NFL) theorem[18], no single algorithm can consistently outperform others across every class of problems without employing domain-specific knowledge. Consequently, practitioners are encouraged to utilize a broad collection of metaheuristic optimization methods, each tailored to different structural properties of problems and diverse performance goals.

Recent research has increasingly focused on integrating XAI, deep learning, and edge-aware architectures for anomaly detection and security in IoT and cyber-physical systems. For example, paper[19] proposed an XAI-enhanced adversarially resilient deep learning framework for secure and transparent edge deployment. In subsequent works, attention-based privacy-preserving models and memory-augmented autonomous agents were introduced to improve robustness and zero-day attack detection capabilities in IoT and Industrial IoT environments[20,21].

In addition, hybrid optimization and learning-based approaches have been explored for intrusion detection tasks. For instance, the integration of genetic algorithms with deep neural networks and edge-aware architectures has demonstrated improved detection performance[22], while ensemble and transfer learning-based XAI models have been applied for detecting zero-day botnet attacks[23]. Metaheuristic-driven classification approaches have also been investigated for critical infrastructure protection, highlighting the effectiveness of combining optimization techniques with machine learning models[24].

While these studies primarily focus on network intrusion detection and IoT-based security scenarios, the present work addresses anomaly detection in cloud system logs, emphasizing the role of NLP techniques and gradient boosting models. Furthermore, this study provides a comparative evaluation of multiple text representation methods in conjunction with metaheuristic optimization, offering complementary insights to existing research in intelligent anomaly detection.

A substantial count of optimization methodologies have been modeled after mechanisms observed in nature, with particular emphasis on evolutionary processes and collective behaviors found in biological systems. Among the most prominent examples are the genetic algorithm (GA)[25], artificial bee colony (ABC)[26], variable neighborhood search (VNS)[27], and particle swarm optimization (PSO)[13], where the latter served as the core optimizer in this work. The field has continued to expand with the introduction of increasingly advanced metaheuristics, including the salp swarm algorithm (SSA)[28], whale optimization algorithm (WOA)[29], bat algorithm (BA)[30], brain storm optimization (BSO)[31], reptile search algorithm (RSA)[32], and chimp optimization algorithm (ChOA)[33]. Collectively, these approaches enhance the adaptability and scope of optimization practices across diverse application areas. In addition, recent innovations such as the sinh cosh optimizer (SCHO)[34] and COLSHADE[35] move away from exclusively bio-inspired paradigms, instead grounding their designs in mathematical principles to achieve more efficient and dependable search performance. A set of the above-mentioned algorithms were also evaluated in the comparative analysis carried out in this research.

The effectiveness of these optimization algorithms has been repeatedly showcased across a variety of practical contexts, with hyperparameter adjustment for ML models standing out as one of their most prominent recent application domains[36]. Nevertheless, their influence reaches far beyond model tuning, shaping progress in numerous disciplines. In healthcare[37,38], metaheuristics contribute to improved diagnostic precision and the design of individualized treatment strategies. In the financial sector[39], they enhance forecasting accuracy, streamline auditing procedures, and reinforce risk management. Moreover, hybrid approaches that integrate several optimization paradigms have emerged as particularly effective, delivering flexible and resilient solutions for complex computational challenges[5,40,41]. Collectively, these developments highlight the adaptability and robustness of contemporary metaheuristic techniques in tackling demanding, data-driven problems across a wide spectrum of high-impact domains[42–44].

Although metaheuristic optimizers have been widely applied across domains, their integration with classification approaches and NLP techniques remains insufficiently investigated within the system log analysis field. In particular, the structured exploration of varied NLP-based feature extraction strategies for anomaly detection in compound log datasets is still largely absent from current research. This gap opens the door for advancing innovative methodologies that couple machine learning with metaheuristic optimization and a diverse array of preprocessing techniques, ultimately offering more powerful and adaptive solutions to the multifaceted challenges inherent in cloud-centric infrastructures.

It is important to note that the contribution of this study does not lie in introducing new embedding techniques, but in systematically analyzing how different text representations interact with optimization-driven model tuning. While TF-IDF, BERT, and Word2Vec are well-established methods, their behavior within an optimization-aware classification framework for cloud log anomaly detection remains insufficiently explored.

In particular, this work investigates how the choice of embedding influences model sensitivity to hyperparameter optimization, classification stability, and anomaly detection performance. By evaluating these representations under a unified and optimized framework, the study provides insights that extend beyond routine benchmarking, offering practical guidance for selecting appropriate preprocessing strategies in real-world log analysis scenarios.

TF-IDF

Term frequency–inverse document frequency (TF–IDF)[8] is a long-established quantitative technique in information retrieval and computational linguistics used to assess the relative significance of a word within a single text compared to an entire dataset. The method generates a weighted index that grows with the frequency of the token inside a given document while being reduced if that token occurs repeatedly throughout the collection. Its canonical formulation is presented in Eq. (1).

$$TF - IDF(t, d, D) = tf(t, d) * idf(t, D) \tag{1}$$

In this context, the term frequency (*TF*) corresponds to the unadjusted count of appearances of a term t in document d. The value scales directly with the number of repetitions, so words that appear more often acquire higher *TF* values. On the other hand, the inverse document frequency (*IDF*) quantifies the rarity of a token across the entire document set D. It is derived by dividing the total corpus size by the number of documents in which the term is observed, thereby accentuating infrequent and potentially more informative words, while diminishing the influence of common, low-value terms. The intermediate measure of document frequency is specified in Eq. (2).

$$DF(t, D) = \frac{t(n)}{D(n)} \tag{2}$$

here, D denotes the total count of documents, t indicates the subset containing the target token, and *DF* represents the corresponding frequency ratio. Frequently recurring stopwords such as "a", "the", or "and" appear in nearly every text, offering negligible discriminative capability. Computing *DF* serves to suppress the weight of these ubiquitous elements while drawing attention to words with greater semantic contribution. To avoid disproportionately large values for unseen or nearly absent terms, the *IDF* component employs a logarithmic scaling of *DF*. This transformation normalizes extreme variations, producing a more balanced and interpretable weighting mechanism. The adjusted formula is given in Eq. (3).

$$IDF(t, D) = log(\frac{D(n)}{t(n)}) \tag{3}$$

Consequently, terms that occur frequently within a specific document but seldom across the larger collection are assigned elevated *TF–IDF* scores, reflecting their heightened descriptive importance.

BERT

Bidirectional encoder representations from transformers (BERT)[9] represents a transformer-driven linguistic framework that exploits the self-attention paradigm to construct deep, context-sensitive interpretations of text. Originally proposed by Google researchers, it has rapidly evolved into a foundational architecture within modern NLP, underpinning a broad spectrum of applications such as information retrieval, automated speech recognition, and cross-lingual translation.

Fundamentally, BERT consists of multiple stacked transformer encoders that utilize multi-head attention. This design allows the model to simultaneously attend to different portions of an input sequence, enabling efficient parallel computation while also capturing intricate interdependencies among tokens.

What distinguishes BERT is its bidirectional strategy for encoding textual context. Instead of limiting the flow of information to a single direction, the model integrates signals from both preceding and subsequent tokens during representation learning. This dual-perspective encoding equips BERT with the ability to model nuanced semantic associations, reconstruct sentence-level meaning with heightened precision, and discern subtle linguistic cues. As a result, BERT has demonstrated exceptional performance in tasks requiring detailed understanding, including sentiment classification, machine comprehension, and natural-language-based question answering.

Word2Vec spaCy

Word embeddings are compact, dense vector representations in low-dimensional space that capture both the semantic subtleties and grammatical roles of lexical items. Within this continuous vector landscape, terms with comparable meanings or syntactic behaviors tend to cluster in close proximity, and their interrelations manifest as quantifiable geometric structures. The advent of the Word2Vec framework[10,45] marked a pivotal shift in generating such representations, offering a highly scalable and computationally efficient approach capable of leveraging vast textual corpora.

The Word2Vec methodology is grounded in two streamlined yet remarkably effective neural models. The Continuous Bag-of-Words (CBOW) architecture infers a target term by synthesizing information from its surrounding linguistic context, whereas the Skip-gram variant reverses the process, beginning with a single word and estimating the neighboring words that are most probable.

Scalability to massive vocabularies is achieved through two optimization heuristics. Hierarchical softmax structures the prediction layer as a binary decision tree, thereby minimizing the computational burden required for probability estimation. In parallel, negative sampling accelerates training by restricting weight updates to a limited subset of output nodes during each iteration. Collectively, these mechanisms reduce the complexity from $O(|V|)$ to $O(\log|V|)$, with $|V|$ representing the overall vocabulary cardinality.

A particularly renowned characteristic of Word2Vec is its ability to generate embeddings that preserve stable and intelligible vector relationships, thereby enabling algebraic operations like:

$$\mathrm{v}_{\mathrm{King}} - \mathrm{v}_{\mathrm{Man}} + \mathrm{v}_{\mathrm{Woman}} \approx \mathrm{v}_{\mathrm{Queen}} \tag{4}$$

The Skip-gram framework is particularly effective at modeling semantic associations among words, reaching 61% accuracy on standard word analogy benchmarks. Conversely, the Continuous Bag-of-Words (CBOW) approach demonstrates greater proficiency in capturing syntactic regularities, achieving 64% accuracy in such evaluations[45].

This research utilizes the spaCy Python package[46,47], a production-grade natural language processing library engineered for both efficiency and scalability. SpaCy delivers optimized implementations of fundamental NLP tasks, including token segmentation, part-of-speech identification, entity recognition, dependency structure parsing, lemmatization, and sentence segmentation. Its Cython-powered backend ensures exceptional runtime performance, positioning it as one of the fastest frameworks for handling extensive text corpora.

The toolkit includes a broad suite of pre-trained models for multiple languages and integrates seamlessly with deep learning ecosystems such as PyTorch and TensorFlow, thereby simplifying the development of tailored workflows and neural architectures. Unlike research-centric toolkits, spaCy is oriented toward robustness, ease of deployment, and a coherent API design, making it especially suitable for real-time systems encompassing dialogue engines, personalized recommendation pipelines, and automated knowledge extraction solutions.

Word2Vec gensim

This work also employed Gensim as NLP component, a widely acknowledged open-source Python framework designed for unsupervised topic discovery and vector-oriented text representation[11]. Gensim proves especially effective for large-scale corpora, as its algorithms are optimized to process data in a streaming fashion rather than loading the entire dataset into memory, consequently offering both scalability and computational efficiency. With robust implementations of Word2Vec, FastText, and related embedding architectures, the library enables transformation of raw log entries into compact numerical vectors that preserve semantic affinities among terms. This sort of representational capacity is crucial for anomaly detection tasks, where capturing contextual dependencies across log messages can markedly enhance classification precision.

Gensim also supplies a wide selection of pretrained embeddings. In this study, a model pretrained on the Google News dataset (default dimensionality: 300) was evaluated. Complementing this, a domain-specific Word2Vec model was trained directly on the target log data corpus to strengthen contextual relevance. To hit a balance between expressive power and computational practicality, particularly for deployment in constrained settings like embedded IoT platforms, a reduced dimensionality of 100 was chosen. This compressed representation, as opposed to the more common 300-dimensional alternative, yielded a much lighter and more easily deployable model while maintaining embedding fidelity. Experimental evaluations demonstrated that this streamlined configuration retained semantic richness and delivered strong classification outcomes.

LightGBM

LightGBM[12], an open-source gradient boosting system developed by Microsoft, was specifically designed to provide exceptional speed and scalability when handling extremely large-scale datasets. Its high-level performance stems from several novel strategies, most notably Gradient-based One-Sided Sampling (GOSS),

which selectively reduces the training sample size while preserving predictive fidelity. Another key innovation, Exclusive Feature Bundling (EFB), merges mutually non-overlapping features, consequently lowering dimensionality and accelerating computation. Collectively, these optimizations enable LightGBM to outperform conventional boosting algorithms in training efficiency, making it particularly beneficial for corpora with millions of entries and high-dimensional attribute spaces. These properties make LightGBM particularly suitable for system log analysis, where datasets are typically large-scale, high-dimensional, and often sparse after text vectorization. In particular, the GOSS mechanism allows the model to focus on more informative instances while reducing computational cost, which is advantageous when processing extensive log data streams. As a result, LightGBM provides an effective balance between computational efficiency and predictive performance in anomaly detection tasks.

Across a wide spectrum of predictive modeling tasks, LightGBM has proven both resilient and precise, with successful deployments in classification, regression, and anomaly detection[48]. Its adaptability has secured adoption in diverse domains encompassing structural engineering[49], financial prediction[50], and fault/defect identification[51]. Furthermore, the framework's architecture supports distributed computation and parallelized execution, ensuring scalability across multi-node infrastructures. Commonly tuned hyperparameters include the leaf count per decision tree (a primary driver of model capacity), the maximum tree depth, and the learning rate or update step size. Each of those parameters exerts a significant influence on the ultimate predictive accuracy and generalization ability of the model.

In comparison to other gradient boosting frameworks such as XGBoost and CatBoost, LightGBM was selected due to its superior computational efficiency and scalability when handling large-scale and high-dimensional datasets. This advantage is primarily attributed to its use of GOSS and EFB, which reduce the number of training instances and effective feature dimensionality without significant loss of information.

These properties are particularly relevant for system log analysis, where text-based feature representations often result in sparse and high-dimensional input spaces. While XGBoost and CatBoost are also powerful boosting methods, LightGBM offers faster training and lower memory consumption in such scenarios, making it more suitable for iterative optimization processes such as metaheuristic hyperparameter tuning employed in this study.

Methods

This section begins with an overview of the baseline PSO algorithm, then introduces the enhanced variant that was subsequently adopted as the primary optimization method in the simulations. Finally, a concise description of the employed classification framework is presented.

Baseline particle swarm optimization

The elementary PSO algorithm is inspired by the collective dynamics of biological swarms and flocks, including the synchronized flight of birds or the schooling patterns of fish[13]. It operates within an *n*-dimensional solution space, where each autonomous agent (called particle in the context of this algorithm) traverses the domain according to predefined update rules, representing possible candidate solutions to the objective (fitness) function. As particles continuously exchange information about the regions they explore, the procedure embodies a population-based optimization framework. Each particle retains two critical memory references: its individual best-known location (*pbest*) and the most favorable position discovered by a designated neighbor (*nbest*) during the ongoing iteration. The particle's coordinate vector encodes its present solution estimate, while the swarm collectively maintains the globally best location attained so far (*gbest*).

The process commences with random initialization of particle positions and velocities within the permitted bounds of the observed search realm. Subsequently, the swarm undergoes iterative refinement, as the particle trajectories are systematically adjusted toward both their own historical best (*pbest*) and the global optimum (*gbest*), resulting in strong collaborative exploration and exploitation. The generation of coordinates at initialization, as well as during subsequent updates, adheres to the canonical PSO formulation[13].

As the optimization procedure progresses, every particle dynamically revises its spatial coordinates based on both its previously achieved best outcomes and its current momentum vector. The velocity and positional updates are governed by Eq. (5), with an explicit velocity constraint defined in Eq. (6).

$$v_{i,d}(t) = w \times v_{i,d}(t-1) + c_1 \times rnd_1 \times (p_{i,d}(t-1) - x_{i,d}(t-1)) + \\ +c_2 \times rnd_2 \times (p_{g,d} - x_{i,d}(t-1)) \tag{5}$$

$$v_{i,d} \epsilon [-v_{max}, v_{max}] \tag{6}$$

here, the instantaneous velocity of the *i*-th particle along the *d*-th axis at iteration *t* is computed. The acceleration coefficients c_1 and c_2 determine the respective influence of cognitive learning (self-reliance on personal best) and social learning (guidance from the neighborhood or global best). The inertia constant *w* moderates the balance between exploration (diverse search of the broader landscape) and exploitation (local refinement around promising regions). Random multipliers rnd_1 and rnd_2, sampled from a uniform distribution in [0, 1], inject randomness into the update mechanism. To prevent excessive displacement, velocities are constrained to lie within $[-v_{max}, v_{max}]$, where v_{max} defines a maximum permissible step size. This procedure ensures trajectories remain bounded and feasible relative to the search domain.

By utilizing collective knowledge-sharing among particles and the iterative reinforcement of successful positional updates, canonical PSO effectively converges toward high-quality and near-optimal solutions. Importantly, it achieves this efficiency without resorting to brute-force search, thereby offering a computationally viable strategy for tackling large-scale, real-world optimization problems.

Proposed modified PSO

Although originally introduced nearly thirty years ago, the canonical PSO algorithm continues to be regarded as a reliable and versatile metaheuristic for addressing a wide spectrum of contemporary optimization tasks[52–54]. Its lasting influence underscores the robustness of its foundational design. Nonetheless, recent comparative studies on advanced benchmarking environments like CEC2022[55] have exposed several limitations, most notably, insufficient exploratory strength during the early stages of the search. To mitigate these issues, the present work introduces an adaptive PSO extension that incorporates evolutionary mechanisms inspired by genetic algorithm (GA)[25], with the goal of enhancing efficiency in complex and dynamically changing landscapes.

The first enhancement is targeted at improving population heterogeneity from the very beginning of the optimization run. This is achieved by embedding the Quasi-Reflective Learning (QRL) technique[56] into the initialization phase of the swarm. In this scheme, the starting population is partitioned into two subsets: one generated through the standard PSO procedure, and the other created via a QRL-driven diversification process designed to increase solution spread in the initial iterations. The additional candidates are produced as quasi-reflective counterparts of the original particles, according to the formulation specified in Eq. (7):

$$X_j^{qr} = rnd\left(\frac{lb_j + ub_j}{2}, x_j\right) \tag{7}$$

In the above formulation, $\frac{lb_j+ub_j}{2}$ corresponds to the midpoint between the lower and upper bounds of the *j*-th search dimension, while *rnd*() denotes a stochastic generator that outputs a random value within given interval.

The second improvement is introduced as follows. During the exploratory phase of the run, covering the first $T/2$ iterations, the algorithm prioritizes diversification of the search. To achieve this, the particle exhibiting the poorest fitness (the least effective solution) is substituted with a new candidate produced via uniform crossover between two randomly chosen individuals—an operation inspired by the GA framework[25].

As the computation advances into the later phase of execution, emphasis transitions toward exploitation. In this stage, the weakest particle is replaced by a newly generated solution obtained from a crossover between the globally best-performing individual and a randomly selected peer from the swarm. This dual-phase strategy preserves a balance, as it promotes thorough refinement of promising regions of the search space while simultaneously sustaining population diversity through stochastic recombination.

The third refinement incorporated to the PSO framework is a soft rollback mechanism introduced in this work. Its purpose is to counter premature convergence and stagnation, activating whenever the swarm exhibits no observable improvement over a span of $T/3$ iterations, where T denotes the maximum iteration count. This cutoff value was determined empirically based on preliminary experimentation, where it provided a suitable balance between allowing sufficient exploration and preventing prolonged stagnation. Although similar threshold-based strategies are commonly used in metaheuristic optimization, the exact proportion ($T/3$ in this case) is problem-dependent and not a fixed standard. Additional threshold values were considered during preliminary analysis. Nevertheless, $T/3$ consistently yielded stable and reliable convergence behavior across the evaluated configurations. Once stagnation is detected, the swarm configuration is rolled back to the most recent state associated with successful progress.

To implement this functionality, two supplementary monitoring variables are defined: the stagnation counter s_count and the stagnation threshold s_tresh, initially set to $s_count = 0$ and $s_tresh = T/3$. At every iteration, if no fitness improvement occurs, the counter s_count is incremented. When s_count reaches or surpasses s_tresh , the reversion process is invoked.

This rollback step incorporates an elitist preservation scheme. Specifically, the best-performing particle (the solution with the finest fitness) is retained, while the remainder of the swarm is reinitialized using the same initialization routine described earlier. This ensures renewed population diversity without discarding the strongest solution discovered thus far.

By embedding this mechanism alongside the previously described enhancements, the modified optimizer is designated as adaptive iteration stagnation-aware PSO (AISAPSO). A procedural breakdown of AISAPSO's workflow is presented in Algorithm 1. Importantly, AISAPSO preserves the original hyperparameter settings and the canonical position–velocity update rules defined in the conventional PSO algorithm[13].

Define maximum population size N
Initialize stagnation threshold parameter s_tresh
Construct $\frac{N}{2}$ particles using the standard PSO initialization scheme
Generate the remaining $\frac{N}{2}$ particles via the QRL diversification strategy
while $(t < T)$ **do**
  Assess the fitness values of all candidate solutions
  Apply the canonical PSO update procedure
  if $(t < T/2)$ **then**
    Substitute the lowest-performing individual in P with an offspring produced by crossover between two randomly chosen particles
  else
    Substitute the weakest particle in P with an offspring produced by crossover between the globally best particle and a randomly selected peer
  end if
  Check for stagnation state
  if (Stagnation detected) **then**
    Invoke soft rollback mechanism
  else
    Archive all current solutions as potential rollback candidates
  end if
end while

Algorithm 1. Operational flow of the AISAPSO optimizer

From a computational perspective, the complexity of the proposed AISAPSO algorithm can be expressed in terms of fitness function evaluations (FFEs), which are the dominant cost in hyperparameter optimization. Let N denote the population size and T the maximum number of iterations. In the baseline case, the core optimization loop requires approximately $N \times T$ FFEs, since each particle must be evaluated once per iteration. The proposed method preserves this same order of growth, that is, $O(N \times T)$, because its additional mechanisms including QRL procedure during initialization and crossover-based replacement of the weakest particle do not add supplementary evaluations, while the stagnation-aware soft rollback strategy introduces only a limited number of additional candidate evaluations when stagnation occurs. Therefore, although the proposed algorithm incurs a slightly higher constant-factor overhead than canonical PSO, its asymptotic complexity remains unchanged and scales linearly with both population size and iteration count.

It should be noted that the proposed modification of the PSO algorithm is not intended to replace standard optimization strategies, but to address specific challenges associated with hyperparameter tuning in this study. In particular, the search space induced by different text representations is high-dimensional and potentially non-convex, with complex interactions between parameters that can affect model performance.

Standard approaches such as grid search or random search may become computationally expensive or inefficient in such circumstances, especially when repeated model training is required. Similarly, canonical PSO may experience premature convergence or stagnation when exploring complex search landscapes.

The proposed modifications, including stagnation-aware adaptation and controlled exploration mechanisms, are designed to improve search stability and convergence behavior under these conditions. Therefore, the modified approach should be viewed as an enhancement aimed at improving optimization efficiency and robustness, rather than a strictly necessary component for achieving effective model performance.

Proposed framework

The proposed simulation framework enables systematic comparison of diverse text representation techniques in natural language processing. Among the evaluated methods are TF-IDF, BERT, and Word2Vec (implemented through spaCy and Gensim, using both Google's pre-trained vectors with dimensionality 300 and a custom Gensim model with dimensionality 100). The preprocessing pipeline involves stop-word elimination and lemmatization, after which the transformed dataset of cloud logs is supplied to a LightGBM classifier to conduct the prediction task. To further refine classification performance, hyperparameters are optimized via metaheuristic search strategies, including the novel AISAPSO algorithm. The overall design of this framework is depicted in Fig. 1, while additional implementation details are outlined in " Simulation setup".

Importantly, the computational burden of the proposed optimization framework is concentrated in the offline training phase, where repeated classifier training and validation are required to assess candidate hyperparameter configurations. In practical deployment scenarios, this optimization does not affect real-time inference, since the final selected model is used only for prediction after training has been completed. Consequently, the training time is a one-time offline cost, while the operational phase remains lightweight. Moreover, absolute training time depends strongly on the execution environment, including processor architecture, memory capacity, software stack, and available hardware acceleration, and therefore can vary substantially across platforms.

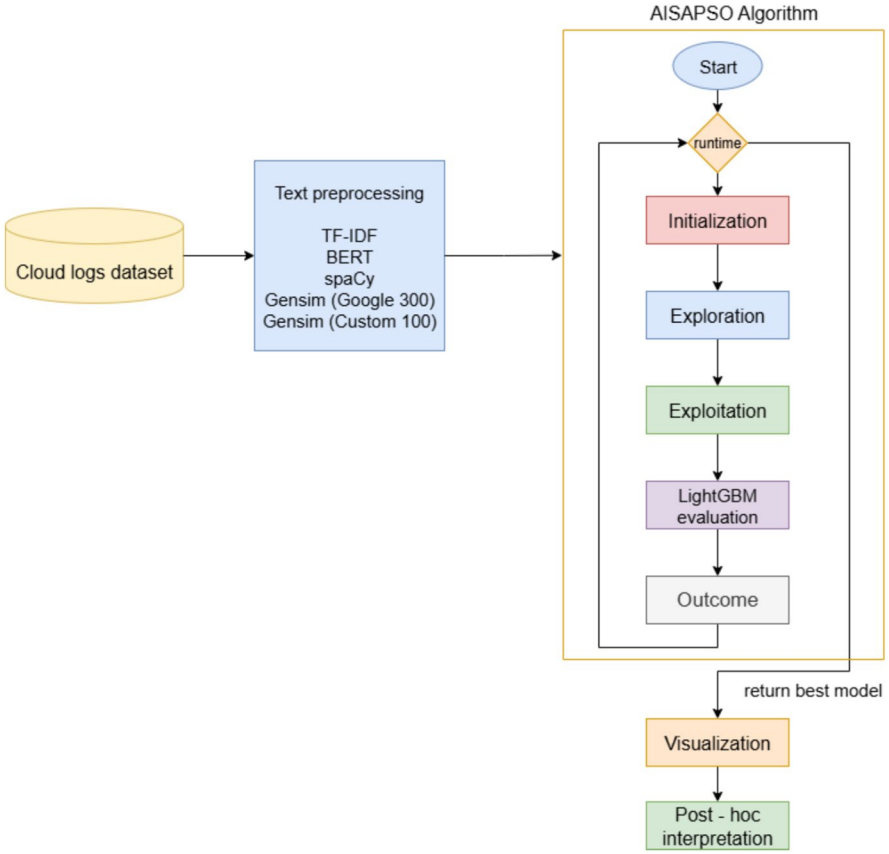


Fig. 1. Flowchart of the proposed optimization framework for evaluating NLP methodologies.

Simulation setup

For experimentation, this work employs a publicly available dataset hosted on Kaggle (https://www.kaggle.com/datasets/krishd123/log-data-for-anomaly-detection). The collection consists of error log records drawn specifically from the Hadoop subset of the broader repository introduced in[57]. To prepare the data for analysis, it was divided into training and testing portions following a 70:30 split, assigning 70% of the samples to model training and reserving 30% for performance assessment. The split was performed in a stratified manner to preserve the original class distribution between normal and anomalous instances.

To prevent any form of data leakage, the training and testing sets were kept strictly disjoint, with no overlap between samples. All preprocessing steps, including text vectorization and feature extraction, were fitted exclusively on the training data and subsequently applied to the testing set. This ensures that no information from the test set was used during model training or hyperparameter optimization. Furthermore, all model selection and hyperparameter optimization procedures were conducted using only the training data, while the test set was used solely for final performance evaluation. An overview of the dataset, including its class distribution, is illustrated in Fig. 2. The dataset is notably imbalanced, with 18,113 normal instances and 1559 anomalous instances, corresponding to approximately a 92% to 8% ratio. This significant imbalance motivates the use of evaluation metrics that are robust to skewed class distributions, such as the Matthews correlation coefficient (MCC), which provides a more reliable assessment than accuracy alone in such settings.

Under a standardized experimental protocol, multiple metaheuristic optimization strategies were benchmarked, each assigned to fine-tune the hyperparameters of the LightGBM classifier. The optimization objective was to enhance the model's capacity for anomaly detection within log data. Selection of the tunable parameters and their search ranges was informed by initial exploratory experiments coupled with sensitivity analyses. It is worth noting that compact, resource-efficient variants of LightGBM were considered to facilitate possible deployment on constrained hardware such as ESP32 microcontrollers. Consequently, the allowable parameter ranges were restricted to smaller bounds. The detailed configuration of these hyperparameter intervals is summarized in Table 1.

It should be noted that the consideration of lightweight configurations in this study is primarily design-oriented, focusing on reducing model complexity and feature dimensionality to enable potential deployment on resource-constrained devices such as ESP32 microcontrollers. Specifically, constraints on hyperparameter ranges and reduced embedding dimensionality (such as 100-dimensional representations) were introduced to limit memory footprint and computational requirements. However, detailed hardware-level evaluation, including measurements of RAM usage, execution time, and energy consumption on embedded platforms, was

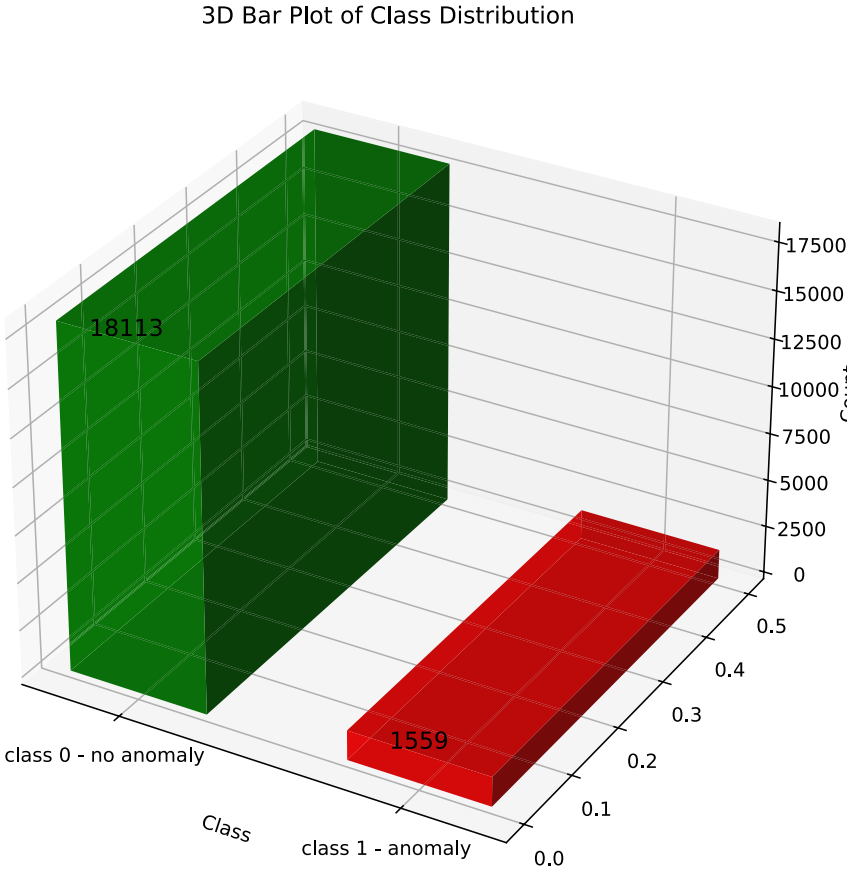


Fig. 2. Dataset and class distribution.

Hyperparameter	Minimum	Maximum
Number of rounds	3	5
Max depth	3	5
Number of leaves	5	10
Min child weight	5	20
Feature fraction	0.1	0.9
Bagging fraction	0.5	1
Min split gain	0.001	0.1
λ L1 regularization	0	5
λ L2 regularization	0	3
Learning rate	0.01	0.9

Table 1. Optimized LightGBM hyperparameters with search limits.

not conducted within the scope of this work. Such analysis would require dedicated deployment and profiling on target hardware and is therefore identified as an important direction for future research.

The performance study encompassed a diverse set of optimization paradigms, combining the newly introduced AISAPSO method with the classical PSO[13]. To establish a fair and representative benchmark, additional metaheuristic algorithms were incorporated, including genetic algorithm (GA)[25], variable neighborhood search (VNS)[27], artificial bee colony (ABC)[26], and the bat algorithm (BA)[30]. The investigation further extended to more contemporary and competitive optimizers, namely the sinh-cosh optimizer (SCHO)[34] and COLSHADE[35]. Collectively, this spectrum of methods provided a robust comparative basis for evaluating LightGBM hyperparameter tuning performance.

To guarantee a fair comparison, every optimization method was re-implemented exclusively for this study, following the baseline parameter configurations prescribed in their respective original works. All algorithms were executed with a fixed population size of eight candidate solutions ($N = 8$) and a maximum of ten iterations

per run ($max\_iter = 10$). For statistical robustness, each approach was repeated independently across thirty separate executions.

As discussed in earlier sections, this study also performed a thorough evaluation of several NLP-oriented preprocessing pipelines, namely TF-IDF, BERT, spaCy, and Gensim. All experiments drew exclusively on the dataset's content column, which was transformed into numerical vectors before being passed to the classifier. For the TF-IDF configuration, the vector space was intentionally constrained to 100 dimensions to align with the objective of producing lightweight models. In contrast, the BERT-based preprocessing utilized only a stratified subset amounting to 5% of the original dataset, thereby retaining class balance while reducing computational overhead. A similarly reduced dataset was applied in the spaCy experiments. In this setup, feature generation relied on the en_core_web_lg language model, where stop words and punctuation were discarded, tokens were lemmatized, and word embeddings were obtained from spaCy's built-in Word2Vec representations. Each token was thus projected into a 300-dimensional vector space, ensuring uniform representation length.

For Gensim preprocessing, two distinct setups were investigated: a model utilizing Google's pre-trained Word2Vec embeddings with the default dimensionality of 300, and a custom Word2Vec model trained specifically for this study, restricted to 100 dimensions to support deployment on resource-constrained hardware such as ESP32 and Arduino-based TinyML systems.

It should be noted that identical dataset sizes and embedding dimensionalities were not strictly enforced across all preprocessing methods. In particular, reduced subsets were used for computationally intensive approaches such as BERT and spaCy, while lower-dimensional representations (including 100-dimensional vectors in TF-IDF and custom Word2Vec) were explored to reflect lightweight deployment scenarios. This design choice was intentional, aiming to evaluate each method under conditions that balance performance and computational feasibility, rather than enforcing uniform configurations that may disadvantage certain approaches. Therefore, the comparison should be interpreted as a practical assessment of different preprocessing strategies under realistic operating conditions.

The performance of the LightGBM classifiers, when paired with the various preprocessing pipelines, was quantified using a conventional set of classification metrics, namely accuracy, precision, recall, and the F1-score, as outlined with Eqs. (8–11)[58]. The primary optimization criterion, however, was the Matthews correlation coefficient (MCC), expressed in Eq. (12)[59], owing to its robustness in evaluating models under class-imbalanced conditions, such as the pronounced imbalance observed in the employed dataset. In addition, the error rate (Eq. 13) was monitored as a supplementary indicator to capture misclassification tendencies. Across all provided formulas, TP and FP correspond to true and false positives, while TN and FN denote true and false negatives. Collectively, this suite of measures establishes a comprehensive evaluation protocol, particularly effective for imbalanced classification scenarios.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Predictions}} \tag{8}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

$$\text{MCC} = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{12}$$

$$\text{Error rate} = 1 - \text{Accuracy} \tag{13}$$

The MCC ranges between –1 and 1, where a score of 1 reflects perfect alignment between predictions and true labels, 0 corresponds to random performance, and –1 signifies a complete inverse relationship. In this study, MCC was adopted as the principal optimization objective, owing to its robustness in handling imbalanced datasets by incorporating all four outcomes of the confusion matrix. To complement this primary metric, the classification error rate (Eq. 13) was also monitored. Although not explicitly optimized, it served as a supportive indicator, providing additional insights into model misclassification tendencies.

Simulation results

TF-IDF simulations

Table 2 presents the comparative results for TF-IDF preprocessing paired with LightGBM models tuned via various metaheuristic optimizers, where evaluation was conducted using MCC as the fitness criterion. Among the contenders, the proposed AISAPSO demonstrated the strongest peak performance, reaching an MCC of .910173 in its best execution, which is a score also matched by SCHO. When considering the lowest-performing executions, COLSHADE achieved the highest baseline with an MCC of .902088. The classical PSO approach delivered the best average performance (.906927), whereas both AISAPSO and PSO shared the highest median result (.908291). In terms of robustness, COLSHADE produced the most consistent outcomes across repeated runs, as indicated by the lowest variance.

Table 3 summarizes the comparative performance of TF-IDF–driven LightGBM models when tuned with different metaheuristic optimizers, where the classification error rate was employed as a complementary

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.910173	.889466	.904599	.908291	.006469	4.18E−05
LGBM-PSO	.908291	.900249	**.906927**	.908291	.002425	5.88E−06
LGBM-GA	.908291	.889466	.903982	.907044	.005893	3.47E−05
LGBM-VNS	.908291	.886655	.899942	.901113	.006769	4.58E−05
LGBM-ABC	.908291	.895770	.900858	.900165	.004467	2.00E−05
LGBM-BA	.908291	.889466	.901158	.902066	.006886	4.74E−05
LGBM-SCHO	.910173	.895770	.904160	.904588	.004337	1.88E−05
LGBM-COLSHADE	.908291	**.902088**	.906405	.907363	**.002289**	**5.24E−06**

Table 2. Fitness results for TF-IDF LightGBM simulations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.014571	.017282	.015266	.014741	.000875	7.66E−07
LGBM-PSO	.014741	.016096	.014961	.014741	.000408	1.67E−07
LGBM-GA	.014741	.017282	.015368	.014910	.000823	6.78E−07
LGBM-VNS	.014741	.017791	.015927	.015757	.000925	8.55E−07
LGBM-ABC	.014741	.016435	.015791	.015842	.000605	3.66E−07
LGBM-BA	.014741	.017282	.015740	.015673	.000926	8.58E−07
LGBM-SCHO	.014571	.016435	.015402	.015419	.000618	3.82E−07
LGBM-COLSHADE	.014741	.015757	.015063	.014910	.000397	1.58E−07

Table 3. Indicator scores for TF-IDF LightGBM simulations.

evaluation metric rather than the primary optimization target. In this analysis, both AISAPSO and SCHO achieved the lowest error rate (.014571), marking the strongest overall outcomes. COLSHADE delivered the most favorable worst-case performance (.015757), while the conventional PSO algorithm reached the best mean error rate (.014961). In terms of median results, AISAPSO and PSO were jointly superior (.014741). Finally, COLSHADE once again demonstrated the highest stability across multiple runs, reflected in the smallest variance.

Table 4 presents a detailed performance breakdown of the top-performing LightGBM classifiers, each fine-tuned with a different metaheuristic optimizer and leveraging TF-IDF features. Among the compared models, AISAPSO and SCHO obtained the strongest results, both reaching an accuracy of .985429 and surpassing the remaining algorithms. Beyond excelling in accuracy, the AISAPSO-optimized model consistently produced balanced results across macro and weighted averages, indicating reliable performance across all target categories. Importantly, this configuration demonstrated particular strength in distinguishing between normal and anomalous log entries, underscoring its value for binary classification within log analysis tasks. Although all optimization strategies yielded competitive outcomes, the uniformly high scores highlight the adaptability of LightGBM to this domain. Nevertheless, AISAPSO set itself apart by combining peak accuracy with well-rounded classification behavior.

To ensure experimental reproducibility, Table 5 details the hyperparameter values selected by each optimization algorithm during the fine-tuning of the top-performing LightGBM classifiers built on TF-IDF features. By explicitly listing the parameter configurations derived from the various metaheuristic techniques, the table enhances methodological transparency and facilitates replication in subsequent studies.

BERT simulations

Table 6 reports the comparative performance of LightGBM classifiers trained on BERT-based feature representations and tuned with various metaheuristic optimization techniques, evaluated through the MCC metric. Within this benchmark, the proposed AISAPSO method emerged as the top performer, achieving the highest MCC of .998839 in its best run (a value also matched by COLSHADE). In addition to this peak score, AISAPSO delivered the strongest results across other indicators, recording the best worst-case outcome (.995351) as well as the highest mean (.996748) and median (.996520) values. In contrast, ABC stood out for its stability, exhibiting the lowest variability across multiple runs.

Table 7 presents a comparative evaluation of LightGBM classifiers trained with BERT-derived features and optimized through different metaheuristic approaches. In this context, the classification error rate was used solely as a supplementary indicator, not as the primary optimization criterion. The AISAPSO strategy delivered the strongest performance overall, achieving the lowest observed error rate of .000169 (along with COLSHADE). Additionally, AISAPSO distinguished itself through consistent reliability, attaining the best worst-case result (.000678) alongside the most favorable mean (.000474) and median (.000508) values across multiple trials. ABC, on the other hand, demonstrated the greatest stability, reflected in its lowest variance among the tested methods. It is important to note that the error rate was not explicitly optimized during training but instead applied as an auxiliary measure to enrich the overall assessment of classifier behavior.

Approach	Metric	Normal	Anomalous	Accuracy	Macro ave	Weight ave
LGBM-AISAPSO	Precision	.999067	.851103	.985429	.925085	.987334
	Recall	.985094	.989316	.985429	.987205	.985429
	f1-score	.992031	.915020	.985429	.953525	.985925
LGBM-PSO	Precision	.998508	.853432	.985259	.925970	.987004
	Recall	.985462	.982906	.985259	.984184	.985259
	f1-score	.991942	.913605	.985259	.952773	.985730
LGBM-GA	Precision	.998508	.853432	.985259	.925970	.987004
	Recall	.985462	.982906	.985259	.984184	.985259
	f1-score	.991942	.913605	.985259	.952773	.985730
LGBM-VNS	Precision	.998508	.853432	.985259	.925970	.987004
	Recall	.985462	.982906	.985259	.984184	.985259
	f1-score	.991942	.913605	.985259	.952773	.985730
LGBM-ABC	Precision	.998508	.853432	.985259	.925970	.987004
	Recall	.985462	.982906	.985259	.984184	.985259
	f1-score	.991942	.913605	.985259	.952773	.985730
LGBM-BA	Precision	.998508	.853432	.985259	.925970	.987004
	Recall	.985462	.982906	.985259	.984184	.985259
	f1-score	.991942	.913605	.985259	.952773	.985730
LGBM-SCHO	Precision	.99067	.851103	.985429	.925085	.987334
	Recall	.985094	.989316	.985429	.987205	.985429
	f1-score	.992031	.915020	.985429	.953525	.985925
LGBM-COLSHADE	Precision	.998508	.853432	.985259	.925970	.987004
	Recall	.985462	.982906	.985259	.984184	.985259
	f1-score	.991942	.913605	.985259	.952773	.985730
	Support	5434	468			

Table 4. TF-IDF LightGBM best-synthesized models detailed metrics.

Approach	Rounds	Max_depth	Leaves	mcw	ff	bf	msg	λ L1	λ L2	lr
LGBM-AISAPSO	5	5	10	10	.9	.8127	.0452	1.6168	1.0895	.7649
LGBM-PSO	4	5	6	6	.5254	.8470	.0940	.0211	2.5532	.8462
LGBM-GA	5	5	5	5	.4629	.5	.0061	.3247	.9269	.9
LGBM-VNS	5	5	9	5	.6530	.5	.1	0	3	.5605
LGBM-ABC	5	5	7	5	.5291	.8469	.0476	0	3	.6274
LGBM-BA	5	4	10	6	.4938	.6522	.1	.3235	2.5811	.6165
LGBM-SCHO	5	5	10	10	.9	1	.0545	1.0672	.2061	.7938
LGBM-COLSHADE	5	5	8	7	.3946	.9339	.0414	1.0756	1.3434	.8675

Table 5. Best-performing TF-IDF LightGBM models hyperparameters configurations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.998839	**.995351**	**.996748**	**.996520**	.001356	1.84E−06
LGBM-PSO	.997679	.991855	.994889	.994199	.001662	2.76E−06
LGBM-GA	.997679	.990717	.995355	.995361	.002141	4.58E−06
LGBM-VNS	.996523	.984858	.993961	.994787	.003332	1.11E−05
LGBM-ABC	.995351	.991855	.993374	.993607	**.001171**	**1.37E−06**
LGBM-BA	.997679	.984877	.994194	.994778	.003294	1.08E−05
LGBM-SCHO	.997677	.993021	.994778	.994204	.001579	2.49E−06
LGBM-COLSHADE	.998839	.991886	.995121	.995351	.001705	2.91E−06

Table 6. Fitness scores for BERT LightGBM simulations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.000169	.000678	.000474	.000508	.000198	3.90E−08
LGBM-PSO	.000339	.001186	.000746	.000847	.000242	5.86E−08
LGBM-GA	.000339	.001355	.000678	.000678	.000312	9.76E−08
LGBM-VNS	.000508	.002203	.000881	.000762	.000484	2.34E−07
LGBM-ABC	.000678	.001186	.000966	.000932	.000170	2.90E−08
LGBM-BA	.000339	.002203	.000847	.000762	.000479	2.30E−07
LGBM-SCHO	.000339	.001017	.000762	.000847	.000230	5.31E−08
LGBM-COLSHADE	.000169	.001186	.000712	.000678	.000249	6.20E−08

Table 7. Indicator scores for BERT LightGBM simulations.

Approach	Metric	Normal	Anomalous	Accuracy	Macro ave	Weight ave
LGBM-AISAPSO	Precision	1	.999816	.999831	.999908	.999831
	Recall	.997863	1	.999831	.998932	.999831
	f1-score	.998930	.999908	.999831	.999419	.999830
LGBM-PSO	Precision	.997863	.999816	.999661	.998840	.999661
	Recall	.997863	.999816	.999661	.998840	.999661
	f1-score	.997863	.999816	.999661	.998840	.999661
LGBM-GA	Precision	.997863	.999816	.999661	.998840	.999661
	Recall	.997863	.999816	.999661	.998840	.999661
	f1-score	.997863	.999816	.999661	.998840	.999661
LGBM-VNS	Precision	.995736	.999816	.999492	.997776	.999492
	Recall	.997863	1	.999492	.998748	.999492
	f1-score	.996798	.999724	.999492	.998261	.999492
LGBM-ABC	Precision	1	.999264	.999322	.999632	.999323
	Recall	.991453	1	.999322	.995726	.999322
	f1-score	.995708	.999632	.999322	.997670	.999321
LGBM-BA	Precision	.997863	.999816	.999661	.998840	.999661
	Recall	.997863	.999816	.999661	.998840	.999661
	f1-score	.997863	.999816	.999661	.998840	.999661
LGBM-SCHO	Precision	1	.999632	.999661	.999816	.999661
	Recall	.995726	1	.999661	.997863	.999661
	f1-score	.997859	.999816	.999661	.998837	.999661
LGBM-COLSHADE	Precision	1	.999816	.999831	.999908	.999831
	Recall	.997863	1	.999831	.998932	.999831
	f1-score	.998930	.999908	.999831	.999419	.999830
	Support	5434	468			

Table 8. BERT LightGBM best-synthesized models detailed metrics.

Table 8 provides an in-depth evaluation of leading LightGBM classifiers, each fine-tuned by a distinct metaheuristic method and trained using BERT-based feature representations. In terms of accuracy, the AISAPSO-optimized model reached the top score of .999831, a performance equaled only by COLSHADE in this setting. In addition, AISAPSO consistently outperformed alternatives on both macro and weighted-average indicators, highlighting its well-balanced predictive strength across classes. The model also excelled at correctly separating normal from anomalous instances, underscoring its reliability for high-stakes binary anomaly detection. Nonetheless, it should be noted that the other optimization strategies also produced remarkably strong results, demonstrating the overall robustness of the approach.

For the sake of reproducibility, Table 9 records the hyperparameter settings determined by each metaheuristic method when optimizing the top-performing LightGBM classifiers built on BERT-derived features. By providing the exact parameter values selected through these optimization processes, the table enhances methodological clarity and offers a foundation for future researchers to replicate the experiments or extend this work.

Word2Vec spaCy simulations

Table 10 provides a head-to-head evaluation of LightGBM classifiers built on spaCy-derived feature sets and fine-tuned with different metaheuristic optimizers, using MCC as the primary performance indicator. Among the contenders, the AISAPSO variant reached the highest observed MCC of .891366 in its best trial, a score also achieved by multiple other optimizers. Beyond peak accuracy, AISAPSO consistently ranked among the

Approach	Rounds	Max_depth	Leaves	mcw	ff	bf	msg	λ L1	λ L2	lr
LGBM-AISAPSO	5	4	7	6	.4226	.5651	.0370	0	0	.7583
LGBM-PSO	5	5	10	5	.3591	.5751	.0898	0	.7126	.7420
LGBM-GA	5	5	10	6	.4096	.5176	.0658	.0499	.3558	.9
LGBM-VNS	5	4	10	5	.9	1	.1	.4799	3	.9
LGBM-ABC	5	4	7	8	.4290	.9023	.0724	1.7586	.3842	.5679
LGBM-BA	4	5	9	6	.3731	.5	.1	0	1.4281	.9
LGBM-SCHO	5	5	8	6	.2380	.5	.001	.2626	3	.8921
LGBM-COLSHADE	5	4	10	6	.3037	.9735	.001	.0046	1.1830	.7021

Table 9. Best-performing BERT LightGBM models hyperparameters configuraitons.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.891366	.888844	.890101	.890105	.000564	3.18E-07
LGBM-PSO	.890105	.888805	.889849	.890105	.000513	2.63E-07
LGBM-GA	.890105	.888805	.889849	.890105	.000513	2.63E-07
LGBM-VNS	.891366	.890105	.890231	.890105	.000378	1.43E-07
LGBM-ABC	.890105	.888844	.889853	.890105	.000505	2.55E-07
LGBM-BA	.891366	.888844	.890105	.890105	.000564	3.18E-07
LGBM-SCHO	.891366	.890105	.890231	.890105	.000378	1.43E-07
LGBM-COLSHADE	.891366	.888844	.890105	.890105	.000564	3.18E-07

Table 10. Fitness scores for Word2Vec spaCy LightGBM simulations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.015249	.015588	.015419	.015419	7.58E−05	5.74E−09
LGBM-PSO	.015419	.015588	.015452	.015419	6.78E−05	4.59E−09
LGBM-GA	.015419	.015588	.015452	.015419	6.78E−05	4.59E−09
LGBM-VNS	.015249	.015419	.015402	.015419	5.08E−05	2.58E−09
LGBM-ABC	.015419	.015588	.015452	.015419	6.78E−05	4.59E−09
LGBM-BA	.015249	.015588	.015419	.015419	7.58E−05	5.74E−09
LGBM-SCHO	.015249	.015419	.015402	.015419	5.08E−05	2.58E−09
LGBM-COLSHADE	.015249	.015588	.015419	.015419	7.58E−05	5.74E−09

Table 11. Indicator scores for Word2Vec spaCy LightGBM simulations.

strongest, securing the excellent average (.890101), median (.890105), and minimum (.888844) results, which underlines both its resilience and reliability. In terms of stability, however, VNS and SCHO stood out by yielding the smallest variance, highlighting their ability to deliver the most consistent outcomes across independent runs.

Table 11 outlines the comparative outcomes of LightGBM classifiers built on spaCy-derived representations and tuned with various metaheuristic optimizers, where the classification error rate was used only as a supporting diagnostic rather than the central optimization criterion. Within this evaluation, the AISAPSO variant delivered the minimum error value of .015249, a benchmark that was also equaled by several other optimizers. Beyond this best-case performance, AISAPSO consistently yielded strong mean, median, and worst-run error results, highlighting its robustness and reproducibility across multiple trials. Meanwhile, the VNS and SCHO metaheuristics achieved the lowest variance, pointing to particularly stable behavior. It is important to note, however, that error rate functioned strictly as a complementary assessment metric and was not the direct target of the optimization process.

Table 12 provides an in-depth assessment of the top LightGBM classifiers, each refined using a distinct metaheuristic strategy and trained on spaCy-derived representations. In terms of classification accuracy, the AISAPSO-driven model reached the highest value of .984751, a result also attained by multiple competing optimizers. Beyond raw accuracy, the AISAPSO configuration displayed well-balanced outcomes across macro and weighted-average measures, highlighting its consistent behavior across both target categories. The model also achieved high precision when differentiating between normal and anomalous instances, underscoring its robustness for binary anomaly detection. Overall, all metaheuristic approaches generated highly competitive and effectively tuned models under this experimental setup.

To promote reproducibility, Table 13 provides a structured overview of the hyperparameter configurations obtained by each optimization technique when refining the top-performing LightGBM models trained on

Approach	Metric	Normal	Anomalous	Accuracy	Macro ave	Weight ave
LGBM-AISAPSO	Precision	.983707	1	.984751	.991854	.985000
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-PSO	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
LGBM-GA	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
LGBM-VNS	Precision	.983707	1	.984751	.991854	.985000
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-ABC	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
LGBM-BA	Precision	.983707	1	.984751	.991854	.985000
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-SCHO	Precision	.983707	1	.984751	.991854	.985000
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-COLSHADE	Precision	.983707	1	.984751	.991854	.985000
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
	Support	5434	468			

Table 12. Word2Vec spaCy LightGBM best-synthesized models detailed metrics.

Approach	Rounds	Max_depth	Leaves	mcw	ff	bf	msg	λ L1	λ L2	lr
LGBM-AISAPSO	4	5	9	5	.1520	.5	.1	2.0295	1.8917	.9
LGBM-PSO	5	5	5	20	.7871	.5	.1	0	2.0653	.9
LGBM-GA	5	3	8	17	.8890	.5712	.0186	5	1.0751	.8926
LGBM-VNS	5	5	5	16	.9	1	.1	.8274	3	.9
LGBM-ABC	5	4	9	5	.8578	.6071	.0221	0	3	.6118
LGBM-BA	5	5	5	16	.9	1	.1	1.1513	3	.9
LGBM-SCHO	5	5	5	5	.8785	.7517	.001	.8798	0	.6650
LGBM-COLSHADE	5	3	8	5	.9	1	.0119	0	0	.9

Table 13. Best-performing Word2Vec spaCy LightGBM models hyperparameters configurations.

spaCy-based features. By documenting the precise parameter values selected through the various metaheuristic approaches, the table enhances methodological transparency and supports future studies seeking to replicate or build upon these results.

Gensim google news pretrained vector size 300

Table 14 presents a comparative assessment of LightGBM classifiers trained on features derived from the Gensim Google News embeddings and refined using various metaheuristic optimizers, with MCC adopted as the central evaluation measure. Within this benchmark, the AISAPSO approach achieved the top MCC score of .891366 in its best run, a result matched by several other methods. More importantly, AISAPSO maintained strong overall performance, producing robust average (.889979), median (.890105), and worst-case (.887581) values, demonstrating both dependability and adaptability. From the standpoint of consistency, however, ABC emerged as the most stable optimizer, attaining the lowest variance and thus showing superior reliability across repeated executions.

Table 15 presents a comparative analysis of LightGBM models trained on Gensim-based feature vectors and optimized through different metaheuristic strategies, with classification error considered only as an auxiliary diagnostic rather than the main optimization objective. In this comparison, the AISAPSO approach attained the lowest observed error of .015249, a value also reached by several alternative optimizers. Beyond this best-case score, AISAPSO further demonstrated solid performance across mean, median, and worst-run outcomes,

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.891366	.887581	.889979	.890105	.000883	7.80E−07
LGBM-PSO	.890105	.887581	.889853	.890105	.000757	5.74E−07
LGBM-GA	.891366	.890105	.890231	.890105	.000378	1.43E−07
LGBM-VNS	.891366	.884932	.889461	.890105	.002004	4.02E−06
LGBM-ABC	.890105	.890105	.890105	.890105	**0**	**0**
LGBM-BA	.891366	.886276	.889848	.890105	.001249	1.56E−06
LGBM-SCHO	.891366	.890105	**.890483**	.890105	.000578	3.34E−07
LGBM-COLSHADE	.891366	.890105	.890231	.890105	.000378	1.43E−07

Table 14. Fitness scores for gensim google news LightGBM simulations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	.015249	.015757	.015435	.015419	.000119	1.41E−08
LGBM-PSO	.015419	.015757	.015452	.015419	.000102	1.03E−08
LGBM-GA	.015249	.015419	.015402	.015419	.000051	2.58E−09
LGBM-VNS	.015249	.016096	.015503	.015419	.000265	7.03E−08
LGBM-ABC	.015419	.015419	.015419	.015419	.000000	0
LGBM-BA	.015249	.015927	.015452	.015419	.000166	2.76E−08
LGBM-SCHO	.015249	.015419	.015368	.015419	.000078	6.03E−09
LGBM-COLSHADE	.015249	.015419	.015402	.015419	.000051	2.58E−09

Table 15. Indicator scores for Gensim Google News LightGBM simulations.

underscoring its reliability and reproducibility. In contrast, the ABC method stood out for delivering the smallest variance, reflecting the most stable behavior across repeated runs. It should be stressed that error rate served purely as a secondary evaluation criterion and was not directly optimized during the training process.

Table 16 presents a comprehensive evaluation of the best-performing LightGBM classifiers, each optimized by a different metaheuristic and trained on features derived from spaCy. With respect to classification accuracy, the AISAPSO-based model delivered the top score of .984751, a performance also matched by several other methods. Importantly, AISAPSO demonstrated balanced effectiveness across both macro- and weighted-average indicators, evidencing reliable performance across the full label set. The model further showed strong precision in separating normal from anomalous events, confirming its robustness in binary anomaly detection. Taken together, the results reveal that all optimization strategies yielded highly competitive and finely tuned classifiers within this experimental framework.

To strengthen reproducibility, Table 17 presents a detailed summary of the hyperparameter settings identified by each optimization method during the tuning of the best LightGBM models trained on Gensim-derived inputs. Recording the exact parameter values produced by the different metaheuristic strategies ensures methodological clarity and provides a solid foundation for future research efforts aimed at replicating or extending these findings.

Gensim custom trained vector size 100

Table 18 presents a comparative study of LightGBM models trained on features derived from a custom Gensim embedding and optimized with various metaheuristic techniques, with MCC employed as the principal evaluation metric. In this benchmark, the AISAPSO-tuned model achieved the maximum attainable MCC of 1 in its best execution, a performance matched only by the SCHO and COLSHADE approaches. Beyond single-run excellence, AISAPSO demonstrated superior consistency, delivering the strongest overall statistics with the highest mean (.998259), median (.997679), and worst-case (.997677) outcomes. This consistency underscores its robustness and reliability across repeated experiments. Moreover, AISAPSO also produced the lowest variance, reinforcing its distinction as the most stable and dependable optimizer among those evaluated.

Table 19 reports a comparative analysis of LightGBM models trained on Gensim-based feature representations and optimized using different metaheuristic techniques, with classification error rate serving as an auxiliary diagnostic rather than the main optimization goal. In this evaluation, the AISAPSO configuration achieved the lowest possible error rate of 0, a result matched only by SCHO and COLSHADE. Beyond this best-case scenario, AISAPSO consistently secured the strongest mean, median, and worst-case error values, underscoring both its robustness and reproducibility across repeated runs. Moreover, AISAPSO delivered the smallest variance, reflecting particularly stable performance across executions. It should be emphasized, however, that error rate in these experiments was strictly a supplementary indicator and not the primary target of the optimization procedure.

Table 20 delivers a comprehensive comparison of the leading LightGBM classifiers, each optimized with a different metaheuristic method and trained on features extracted from a custom Gensim embedding model. Regarding classification accuracy, the AISAPSO-tuned model achieved a flawless score of 100%, a benchmark equally attained by both the SCHO and COLSHADE optimizers. Beyond accuracy alone, AISAPSO produced

Approach	Metric	Normal	Anomalous	Accuracy	Macro ave	Weight ave
LGBM-AISAPSO	Precision	.983707	1	.984751	.991854	.984999
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-PSO	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
LGBM-GA	Precision	.983707	1	.984751	.991854	.984999
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-VNS	Precision	.983707	1	.984751	.991854	.984999
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-ABC	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
LGBM-BA	Precision	.983707	1	.984751	.991854	.984999
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-SCHO	Precision	.983707	1	.984751	.991854	.984999
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
LGBM-COLSHADE	Precision	.983707	1	.984751	.991854	.984999
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
	Support	5434	468			

Table 16. Gensim Google News LightGBM best-synthesized models detailed metrics.

Approach	Rounds	Max_depth	Leaves	mcw	ff	bf	msg	λ L1	λ L2	lr
LGBM-AISAPSO	5	5	9	5	.9	.5	.1	2.1905	0	.9
LGBM-PSO	5	4	7	15	.8145	.9674	.0855	2.3419	.7342	.4919
LGBM-GA	3	5	10	5	.9	.6587	.1	0	.1528	.9
LGBM-VNS	5	5	9	5	.9	1	.1	3.3513	3	.9
LGBM-ABC	4	4	9	8	.3415	.6352	.0555	1.018	1.9616	.6042
LGBM-BA	4	4	7	5	.9	1	.1	.7298	.9701	.9
LGBM-SCHO	5	5	8	5	.9	1	.0012	1.337	0	.9
LGBM-COLSHADE	5	5	10	5	.9	1	.0089	2.9337	.1239	.5425

Table 17. Best-performing Gensim Google News LightGBM models hyperparameters configurations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	1	**.997677**	**.998259**	**.997679**	**.000779**	**6.07E−07**
LGBM-PSO	.998839	.996515	.997562	.997677	.000964	9.30E−07
LGBM-GA	.998839	.996515	.997561	.997677	.000813	6.61E−07
LGBM-VNS	.998839	.994243	.997105	.997677	.001176	1.38E−06
LGBM-ABC	.998839	.993021	.996050	.996515	.001578	2.49E−06
LGBM-BA	.998839	.994204	.996870	.997677	.001640	2.69E−06
LGBM-SCHO	1	.995358	.997562	.997677	.001212	1.47E−06
LGBM-COLSHADE	1	.995370	.997447	.997677	.001248	1.56E−06

Table 18. Fitness scores for Gensim custom trained LightGBM simulations.

Method	Best	Worst	Mean	Median	Std	Var
LGBM-AISAPSO	0	.000339	.000254	.000339	.000114	1.29E−08
LGBM-PSO	.000169	.000508	.000356	.000339	.000141	1.98E−08
LGBM-GA	.000169	.000508	.000356	.000339	.000119	1.41E−08
LGBM-VNS	.000169	.000847	.000424	.000339	.000174	3.01E−08
LGBM-ABC	.000169	.001017	.000576	.000508	.000230	5.28E−08
LGBM-BA	.000169	.000847	.000457	.000339	.000240	5.77E−08
LGBM-SCHO	0	.000678	.000356	.000339	.000177	3.13E−08
LGBM-COLSHADE	0	.000678	.000373	.000339	.000182	3.33E−08

Table 19. Indicator scores for Gensim custom trained LightGBM simulations.

Approach	Metric	Normal	Anomalous	Accuracy	Macro ave	Weight ave
LGBM-AISAPSO	Precision	1	1	1	1	1
	Recall	1	1	1	1	1
	f1-score	1	1	1	1	1
LGBM-PSO	Precision	.999816	1	.999831	.999908	.999831
	Recall	1	.997863	.999831	.998932	.999831
	f1-score	.999908	.998930	.999831	.999419	.999830
LGBM-GA	Precision	.999816	1	.999831	.999908	.999831
	Recall	1	.997863	.999831	.998932	.999831
	f1-score	.999908	.998930	.999831	.999419	.999830
LGBM-VNS	Precision	.999816	1	.999831	.999908	.999831
	Recall	1	.997863	.999831	.998932	.999831
	f1-score	.999908	.998930	.999831	.999419	.999830
LGBM-ABC	Precision	.999816	1	.999831	.999908	.999831
	Recall	1	.997863	.999831	.998932	.999831
	f1-score	.999908	.998930	.999831	.999419	.999830
LGBM-BA	Precision	.999816	1	.999831	.999908	.999831
	Recall	1	.997863	.999831	.998932	.999831
	f1-score	.999908	.998930	.999831	.999419	.999830
LGBM-SCHO	Precision	1	1	1	1	1
	Recall	1	1	1	1	1
	f1-score	1	1	1	1	1
LGBM-COLSHADE	Precision	1	1	1	1	1
	Recall	1	1	1	1	1
	f1-score	1	1	1	1	1
	Support	5434	468			

Table 20. Gensim custom trained LightGBM best-synthesized models detailed metrics.

perfect results on macro and weighted-average evaluations, confirming its uniform effectiveness across all prediction classes. The model further demonstrated ideal precision in distinguishing between normal and anomalous samples, reinforcing its dependability for binary anomaly detection. Notably, every metaheuristic strategy under consideration produced strong, well-calibrated models in this experimental setting.

To strengthen reproducibility, Table 21 presents a systematic summary of the hyperparameter settings produced by each optimization strategy when tuning the best-performing LightGBM models built on custom trained Gensim-derived inputs. Recording the exact parameter choices made by the different metaheuristics not only improves methodological clarity but also provides a concrete foundation for future research aiming to replicate or extend these findings.

Visual comparative analysis across five different NLP methods

Figure 3 displays a detailed comparison of several metaheuristic algorithms used to optimize LightGBM hyperparameters for detecting anomalies in cloud system logs. The evaluation spans multiple text preprocessing strategies: TF-IDF (upper row), BERT embeddings, spaCy embeddings, Gensim with Google News pretrained vectors (300 dimensions) and Gensim trained on a custom corpus (100 dimensions) (lower row). The violin plots summarize the distributions of MCC scores over 30 independent runs, providing a statistically grounded view of optimizer behavior. These visualizations highlight patterns of central tendency, variability, and asymmetry, making visible the balance each method strikes between consistency and exploratory range. Among the tested

Approach	Rounds	Max_depth	Leaves	mcw	ff	bf	msg	λ L1	λ L2	lr
LGBM-AISAPSO	5	5	10	5	.5054	.5928	.0246	0	1.4870	.7824
LGBM-PSO	5	5	10	5	.6639	.7619	.0444	0	1.0016	.9
LGBM-GA	5	5	9	5	.6998	.5508	.0487	1.3503	1.4324	.9
LGBM-VNS	5	5	10	8	.4089	.5	.1	.2269	2.9046	.9
LGBM-ABC	5	5	10	9	.4788	.505	.0495	0	.5407	.8376
LGBM-BA	5	4	9	8	.7907	1	.1	1.5179	.9758	.5845
LGBM-SCHO	5	5	10	5	.3729	1	.001	0	2.4358	.9
LGBM-COLSHADE	5	5	8	5	.6077	.5	.01	.132	1.2624	.9

Table 21. Best-performing Gensim custom trained LightGBM models hyperparameters configurations.

optimizers, the proposed AISAPSO emerges as the most effective, consistently producing the highest best-case outcomes across all preprocessing setups, which is a testament to its strong global search capability. However, the somewhat wider variance seen in certain scenarios underscores its intentional exploration mechanism, designed to avoid premature convergence and promote broader solution discovery.

In contrast, many competing optimizers exhibit much tighter distributions with lower variance, which serves as an indicator of greater reliability, yet also of a susceptibility to settle prematurely in local optima. The convergence curves further support this observation, charting the progression of the objective function across iterations. Algorithms that converge smoothly and rapidly, like proposed AISAPSO, highlight a well-calibrated balance between exploration and exploitation. This effectiveness can be attributed to design elements like stagnation awareness and explicit diversity-preserving mechanisms. By comparison, optimizers whose curves flatten early or plateau altogether reveal a diminished capacity to adapt to the complex structure of the hyperparameter search space.

Collectively, the evidence underscores how algorithmic architecture critically shapes the outcomes of hyperparameter tuning in NLP classification tasks. Design factors like population diversity and stagnation awareness are shown to exert direct influence on convergence speed, solution robustness, and reproducibility across varied feature representations. For practitioners, these findings provide actionable insight: models optimized with attention to these principles are more resilient to initialization sensitivity and variability, enabling reliable, high-performance deployment in production environments.

Figure 4 presents a detailed comparison of classification error distributions and convergence behaviors for multiple metaheuristic approaches used in tuning XGBoost hyperparameters. The evaluation spans five text preprocessing pipelines: TF-IDF (upper row), BERT embeddings, spaCy embeddings, Gensim pretrained on Google News (300 dimensions) and Gensim trained on a domain-specific corpus (100 dimensions) (lower row). The box plots summarize the spread and central tendency of error rates over 30 independent trials, serving as a key diagnostic of how well models generalize under class-imbalanced scenarios. Optimizers that achieve lower medians with compact interquartile ranges stand out as methods capable of not only locating strong solutions but also sustaining stability across random initializations. Among these, AISAPSO consistently reports the lowest error values, highlighting the benefits of its adaptive mechanisms that balance diversification and exploitation. This design enables the algorithm to sidestep local optima while effectively minimizing misclassification in binary anomaly detection.

The convergence plots further reinforce these observations by tracing how error rates decline over successive iterations, exposing clear contrasts in both learning velocity and stability across optimizers. Methods that exhibit fast, steady descent curves demonstrate a carefully tuned equilibrium between exploitation and exploration, which represents an ability particularly critical when navigating the complexity of high-dimensional hyperparameter realms. Conversely, optimizers that plateau early or show erratic trajectories reflect limitations in sustaining population diversity and adapting to shifting search dynamics.

Taken together, these comparative illustrations reveal the fundamental trade-offs embedded in metaheuristic design: broad exploration versus controlled convergence. They also offer practical guidance for choosing optimization strategies tailored to NLP-based classification tasks. By emphasizing not just raw accuracy but also reproducibility, robustness, and interpretability, such insights help ensure that real-world deployments, for instance, in cloud log anomaly detection, can achieve reliable and sustainable performance.

Next, Fig. 5 presents swarm plots comparing how different metaheuristic optimizers distribute and diversify both objective function scores and classification error rates. The assessment spans several preprocessing pipelines: TF-IDF (upper row), BERT embeddings, spaCy embeddings, Gensim pretrained on Google News (300 dimensions), and Gensim trained on a domain-specific corpus (100 dimensions) (lower row). The plots summarize results from the final iteration of the top-performing runs. AISAPSO typically exhibits a compact spread, indicating that nearly all solutions converged tightly around the global best. A similar clustering effect can be observed, to varying degrees, in most of the other optimizers considered.

By simultaneously depicting solution quality and population diversity, these visualizations highlight each algorithm's capacity to maintain heterogeneity, avoid early stagnation, and repeatedly uncover high-quality hyperparameter settings. These features are fundamental to constructing NLP classification models that are not only accurate but also stable and broadly generalizable.

Analysis of the objective function outcomes (where larger values correspond to stronger MCC performance) reveals that the custom-trained Gensim embeddings with a 100-dimensional vector space deliver the most

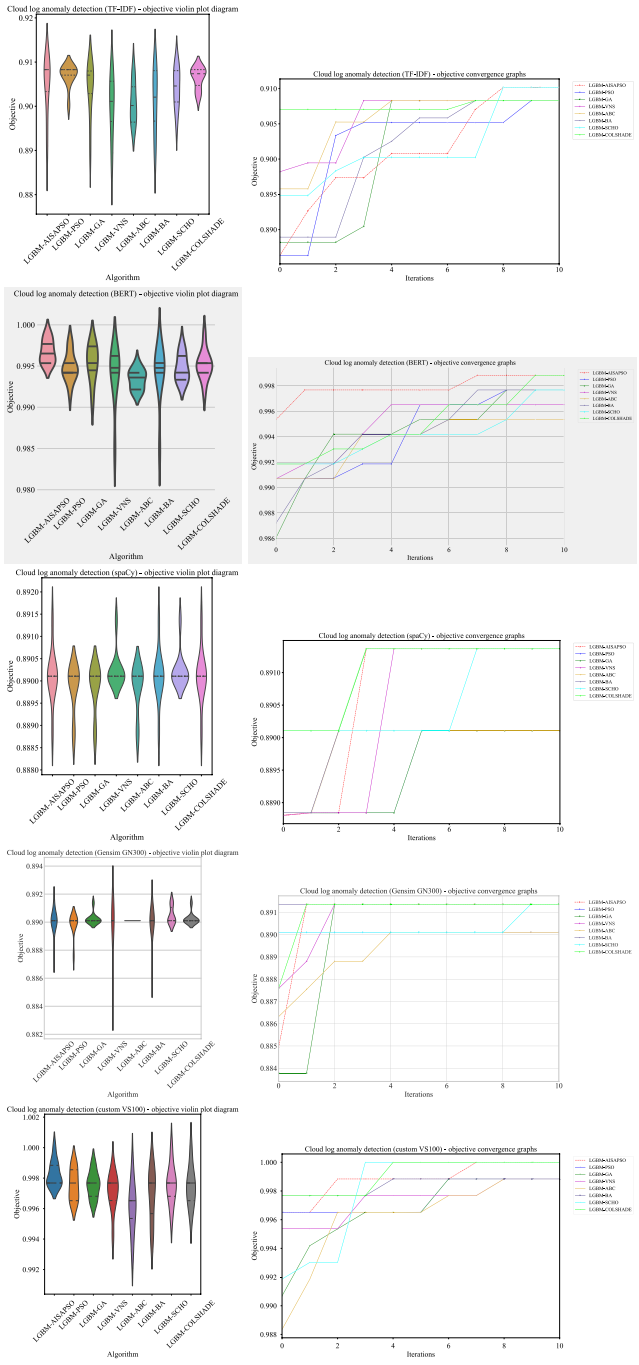


Fig. 3. Fitness function distributions and convergence diagrams for TF-IDF (top), BERT, spaCy, Gensim Google News and Gensim custom (bottom) experiments.

favorable results overall. Compared with BERT, TF-IDF, spaCy Word2Vec and Gensim Google News pipelines, this tailored representation achieves consistently higher medians and superior best-case MCC values, as shown in the violin plots. Its advantage stems from embeddings that are domain-adapted to the structure of cloud system logs, capturing task-specific semantics more effectively than general-purpose models. The convergence profiles further emphasize this point: Gensim-100–based classifiers display stable, reliable trajectories across multiple optimizers, suggesting that this representation facilitates smoother discovery of high-quality hyperparameter configurations and closer alignment between predictions and true labels. These benefits are especially pronounced in AISAPSO-optimized runs, where the Gensim-100 pipeline achieves the strongest MCC values, underscoring its capacity to enhance discriminative performance in this anomaly detection context.

When classification performance is examined through error rates (where lower is better), Gensim-100 again stands out as the most competitive. Its box plots reveal lower medians and tighter spreads, pointing to models that generalize effectively while minimizing misclassifications under class imbalance. While BERT continues

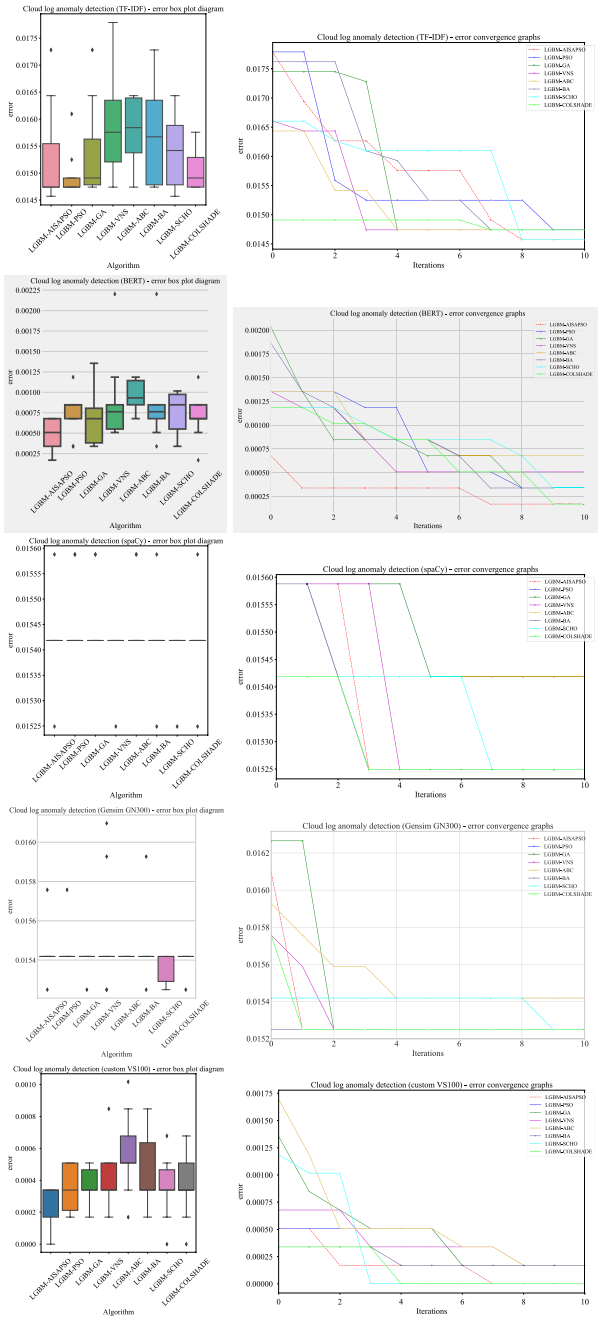


Fig. 4. Error rate distributions and convergence diagrams for TF-IDF (top), BERT, spaCy, Gensim Google News and Gensim custom (bottom) simulations.

to provide strong, reliable performance and TF-IDF or spaCy serve as practical lighter-weight alternatives, the collective evidence positions the custom-trained Gensim embeddings as the optimal choice. Their combination of high MCC values, low error distributions, and robustness across metaheuristic optimizers establishes Gensim-100 as the leading preprocessing strategy for building dependable NLP models in cloud log anomaly detection. Another important point to be taken is that custom-trained Gensim is lightweight, using vector size of just 100, which makes it suitable for possible deployment on resource-constrained microcontroller platforms.

Lastly, Fig. 6 delivers the radar graphs depicting macro and weighted averages, providing a holistic view of classifier performance across multiple evaluation metrics. Macro-averaged scores give equal weight to each class, thereby highlighting how well models handle minority categories that are often underrepresented in log anomaly detection. In contrast, weighted averages emphasize the contribution of majority classes, reflecting overall performance in proportion to class distribution. By visualizing both perspectives side by side, the radar plots reveal important trade-offs: methods that excel in weighted averages may still struggle with minority detection, while those with balanced macro scores demonstrate stronger robustness under imbalance. Together,

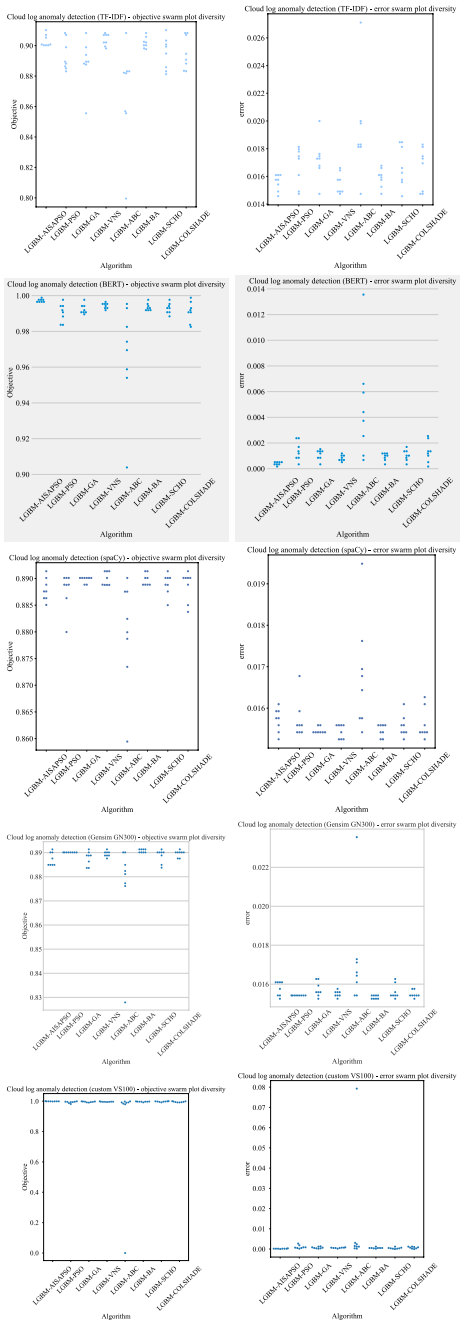


Fig. 5. Swarm plots of both objective and error rate for TF-IDF (top), BERT, spaCy, Gensim Google News and Gensim custom (bottom) simulations.

these representations offer a complementary diagnostic for assessing generalization, fairness, and reliability of the different metaheuristic-optimized models.

Discussion

The proposed framework is motivated by the need to jointly address three key challenges in cloud log anomaly detection: effective representation of unstructured log data, efficient and scalable classification, and reliable hyperparameter optimization in complex search spaces. Rather than treating these aspects independently, this study integrates natural language processing-based feature representations with a gradient boosting model and a metaheuristic optimization strategy within a unified framework.

The significance of this integration lies in enabling a systematic evaluation of how different text representations interact with optimization-driven model tuning, which is not commonly explored in existing literature. This allows the study to move beyond isolated improvements and instead provide insight into the combined effect of preprocessing, model architecture, and optimization on anomaly detection performance.

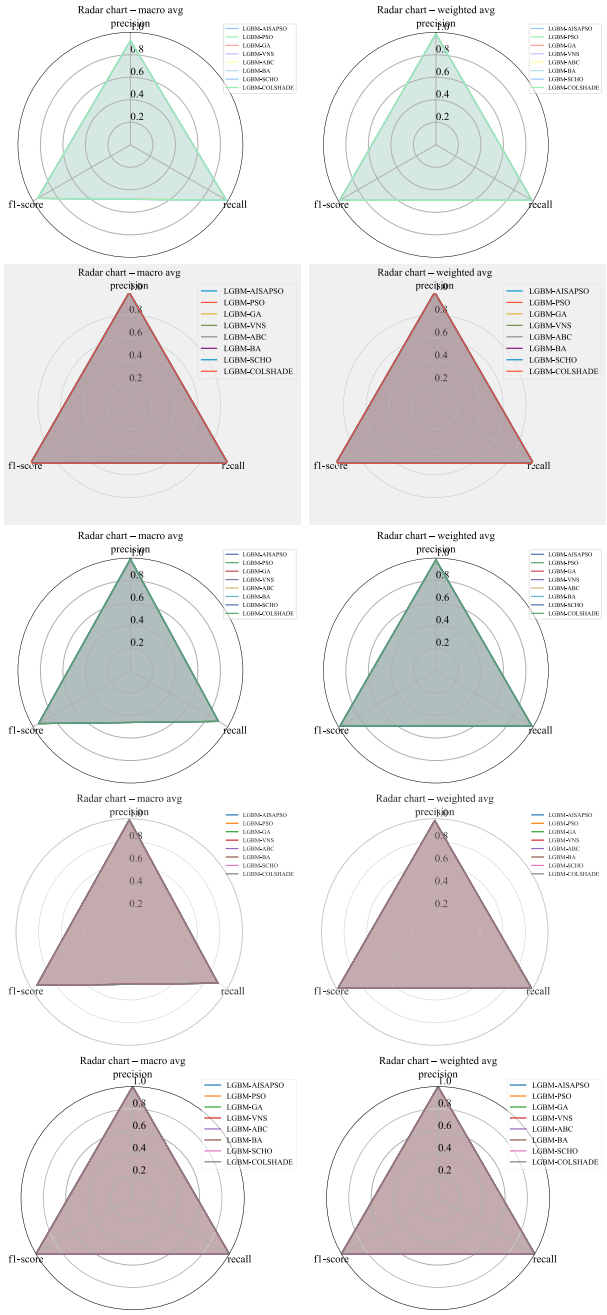


Fig. 6. Radar charts of both macro and weighted averages for TF-IDF (top), BERT, spaCy, Gensim Google News and Gensim custom (bottom) simulations.

Furthermore, the evaluation framework incorporates multiple performance metrics and repeated experimental runs to ensure reliability, while interpretability is enhanced through SHAP-based analysis. This not only supports performance assessment but also provides transparency into model behavior, facilitating practical decision-making in log analysis scenarios. Together, these elements clarify the contribution of the proposed framework as a comprehensive and systematically evaluated approach to cloud log anomaly detection.

In this study, the term robustness is used to refer to the stability and consistency of model performance across different experimental configurations and repeated runs. Specifically, robustness is reflected in the ability of the proposed framework to maintain high performance across multiple text representations and optimization settings, as well as in its consistent results over 30 independent runs.

Additionally, robustness is associated with reliable performance under class imbalance, as demonstrated by strong MCC values and high recall for anomalous instances. It should be noted that robustness in this context does not imply resilience to adversarial manipulation or guaranteed generalization to unseen data distributions, which remain important directions for future investigation.

The near-perfect performance observed in several configurations can be attributed to a combination of factors, including the effectiveness of the preprocessing pipeline, the expressive capability of the LightGBM classifier, and the use of optimized hyperparameters. Although the observed performance is strong, it is important to note that near-perfect accuracy in such settings may also raise concerns regarding potential overfitting or the relative simplicity of the dataset. However, the use of repeated independent runs, multiple evaluation metrics, and consistent performance across configurations suggests that the results reflect genuine model capability rather than memorization of training data.

It is important to clarify that the observed performance improvements do not arise from a single component, but from the combined effect of multiple elements within the proposed framework. In particular, the choice of text representation influences the quality and separability of the input features, the LightGBM classifier provides an efficient and expressive modeling capability, and the optimization strategy contributes to improved hyperparameter selection.

The comparative results indicate that different embeddings lead to varying performance levels, highlighting the importance of feature representation. At the same time, the use of optimization enhances model performance across configurations by refining parameter settings. Therefore, the reported gains should be interpreted as the result of the interaction between preprocessing, model architecture, and optimization, rather than being attributed to a single factor in isolation.

To ensure that these results are not due to random variation, all experiments were conducted using a fixed train-test split (70:30), and each optimization procedure was repeated across 30 independent runs. Furthermore, performance was evaluated using several metrics, including accuracy, precision, recall, F1-score, and MCC, the latter being particularly suitable for imbalanced datasets. Model interpretability was also assessed using SHAP, which provided insight into feature contributions and confirmed that the model decisions were driven by meaningful patterns in the data rather than spurious correlations.

Beyond providing insight into feature contributions, interpretability plays an important role in practical decision-making within anomaly detection systems. In particular, SHAP explanations enable system operators to understand why a specific log entry is classified as anomalous by highlighting the most influential features. This information can support validation of detected anomalies, facilitate root-cause analysis, and help prioritize responses by distinguishing between meaningful anomalies and potential false positives. Furthermore, interpretability enhances trust in the model by providing transparency, which is especially important in security-critical environments where automated decisions must be explainable and auditable.

From an error analysis perspective, most misclassifications are associated with borderline or ambiguous log entries, where the distinction between normal and anomalous behavior is less pronounced. These cases contribute to a small number of false positives and false negatives, particularly affecting the minority anomaly class. Nevertheless, the consistently high recall values for anomalies indicate that the model prioritizes detection of abnormal events, which is desirable in practical cybersecurity and system monitoring applications.

From a practical perspective, improvements in anomaly detection performance can support more effective monitoring of cloud systems, but they do not directly translate into guaranteed improvements in system reliability. In real-world cloud infrastructures, reliability depends on multiple factors, including system architecture, fault tolerance mechanisms, alert handling processes, and response strategies.

In this context, the proposed framework contributes by enhancing the detection of anomalous events in system logs. In particular, high recall for anomalous instances reduces the likelihood of missed critical events, while controlled precision helps manage false alarms and operational overhead. These improvements can support more informed decision-making and faster response within existing monitoring pipelines.

However, the overall impact on system reliability depends on how detected anomalies are integrated with downstream processes such as alert correlation, incident response, and automated mitigation strategies. Therefore, the proposed approach should be viewed as a supporting component within a broader cloud monitoring and security ecosystem rather than a standalone solution for ensuring system reliability.

Despite the strong performance, the results should be interpreted in the context of dataset characteristics. The Hadoop log dataset exhibits relatively structured and well-separated patterns between normal and anomalous entries, which contributes to the high observed accuracy. In addition, the dataset is imbalanced, with anomalous instances comprising a much smaller proportion of the data. In such settings, metrics such as MCC and class-specific recall provide a more informative assessment than accuracy alone. The strong performance across these metrics confirms that the model remains effective despite the imbalance.

However, real-world production environments often involve significantly more heterogeneous and noisy log data, including evolving patterns and previously unseen anomaly types. Under such conditions, classification performance may be lower due to increased ambiguity and distribution shifts between training and deployment data. Therefore, while the proposed framework demonstrates strong effectiveness under controlled conditions, further validation on more diverse and noisy datasets is necessary to fully assess its generalization capability.

Although the proposed framework is designed to be adaptable to different types of log data, its evaluation in this study is limited to the Hadoop dataset. Cloud environments often involve heterogeneous logging sources with varying formats, semantics, and levels of noise, which may influence both feature representation and model performance. Therefore, the general applicability of the framework across diverse cloud systems should be interpreted with caution, and further validation on heterogeneous log datasets is required to fully assess its robustness.

Regarding scalability, the proposed framework is well suited for larger datasets and practical cloud environments because the optimization process is naturally parallelizable at the population level, with fitness evaluations of different candidate solutions being executed independently. In addition, the use of LightGBM contributes to scalability due to its efficiency on high-dimensional data and large sample sizes. Nevertheless, as dataset size and feature dimensionality increase, the primary computational bottleneck remains the repeated

training of candidate models during hyperparameter search. For this reason, the proposed framework is most appropriate for offline model development and periodic retraining, while deployment in production environments can rely on the optimized model without incurring the full optimization cost.

Validation and interpretation

Statistical validation

While side-by-side comparisons of optimization algorithms can provide useful impressions, drawing firm conclusions from a single run is inherently unreliable. Metaheuristic methods are stochastic by design, and their outcomes can differ substantially depending on initialization, making any one execution potentially unrepresentative. To address this variability, the study conducted 30 independent executions for each algorithm, each seeded with a different random initialization. This design produced a wide and balanced set of outcomes, offering a more dependable basis for evaluation. Such replication makes it possible to apply rigorous statistical analyses that separate true performance differences from random noise. By adhering to established best practices for benchmarking metaheuristic algorithms[60], the study strengthens both the robustness and credibility of its comparative findings.

Statistical hypothesis testing is typically grouped into two major categories: parametric and non-parametric approaches. Parametric techniques are generally more powerful but rely on strict assumptions—independence of observations, normality of data distributions, and equality of variances across groups (homoscedasticity)[60]. In this study, independence was ensured by assigning a distinct random seed to every optimizer run. Homoscedasticity was assessed using Levene's test[61], which returned a p-value of .61, indicating no significant variance differences between groups. Normality, the remaining condition, was tested with the Shapiro–Wilk procedure[62]. As shown in Table 22, all results yielded p-values below the conventional .05 threshold, leading to rejection of the normality assumption. This finding confirms that parametric tests would not provide a valid framework for subsequent statistical comparisons.

Additional evidence is provided by the kernel density estimation (KDE) curves in Fig. 7, which clearly illustrate deviations from a Gaussian profile. In light of these observations, the analysis proceeded with non-parametric testing methods, chosen for their ability to deliver reliable assessments without relying on restrictive distributional assumptions.

Since the normality assumption was violated, the evaluation advanced with non-parametric methods. In particular, the Wilcoxon signed-rank test[63] was employed to conduct pairwise comparisons between the proposed AISAPSO and each competing optimizer. The corresponding p-values are reported in Table 23. For experiments using TF-IDF, BERT, spaCy, and the custom-trained Gensim embeddings, all p-values were below the conventional significance threshold of $\alpha = .05$, confirming that AISAPSO achieved statistically significant gains over every alternative approach. This outcome reinforces the conclusion that AISAPSO's superior results are not the product of stochastic variation but reflect a genuine and consistent advantage in the tested settings. The only exception arose in the Gensim Google News 300 experiments, where the p-value for SCHO exceeded the threshold, indicating that AISAPSO significantly outperformed all algorithms except SCHO under this preprocessing configuration.

Comparison with baseline models

For further assessment of the obtained results, the best produced model in each experiment were placed in the comparative analysis against a set of competitive benchmark models. This collection of benchmarks included AdaBoost[64], CatBoost[65], CNN[66], decision tree[67], K-nearest neighbors (KNN)[68], plain LightGBM, multi-layer perceptron (MLP), random forest[69] and XGBoost[70].

In this study, hyperparameter optimization using metaheuristic algorithms was applied specifically to the LightGBM classifier configurations under investigation. Baseline models were evaluated using standard or commonly adopted parameter settings, rather than undergoing the same optimization procedure. This design choice was made to provide a reference point reflecting typical usage scenarios and to highlight the performance gains achieved through the proposed optimization strategy.

The outcomes of these experiments are displayed within Table 24. Although all observed baselines obtained respectable accuracy levels, it is clear that the proposed methodology yielded excellent per-class predictions with slightly higher accuracy.

Although tuning baseline models could potentially improve their performance, the primary objective of this work is to assess the impact of advanced optimization strategies on LightGBM in combination with different preprocessing techniques. Therefore, maintaining baseline configurations without metaheuristic tuning allows clearer isolation of the contribution of the proposed optimization framework.

LightGBM	AISAPSO	PSO	GA	VNS	ABC	BA	SCHO	COLSHADE
TF-IDF	.023	.034	.022	.029	.021	.019	.017	.015
BERT	.029	.021	.023	.026	.022	.018	.028	.025
spaCy	.029	.023	.025	.031	.024	.027	.032	.021
Gensim Google News (300)	.027	.023	.029	.024	.031	.033	.022	.019
Gensim custom (100)	.029	.023	.022	.027	.023	.028	.021	.026

Table 22. Shapiro–Wilk p-values for every algorithm and preprocessing method.

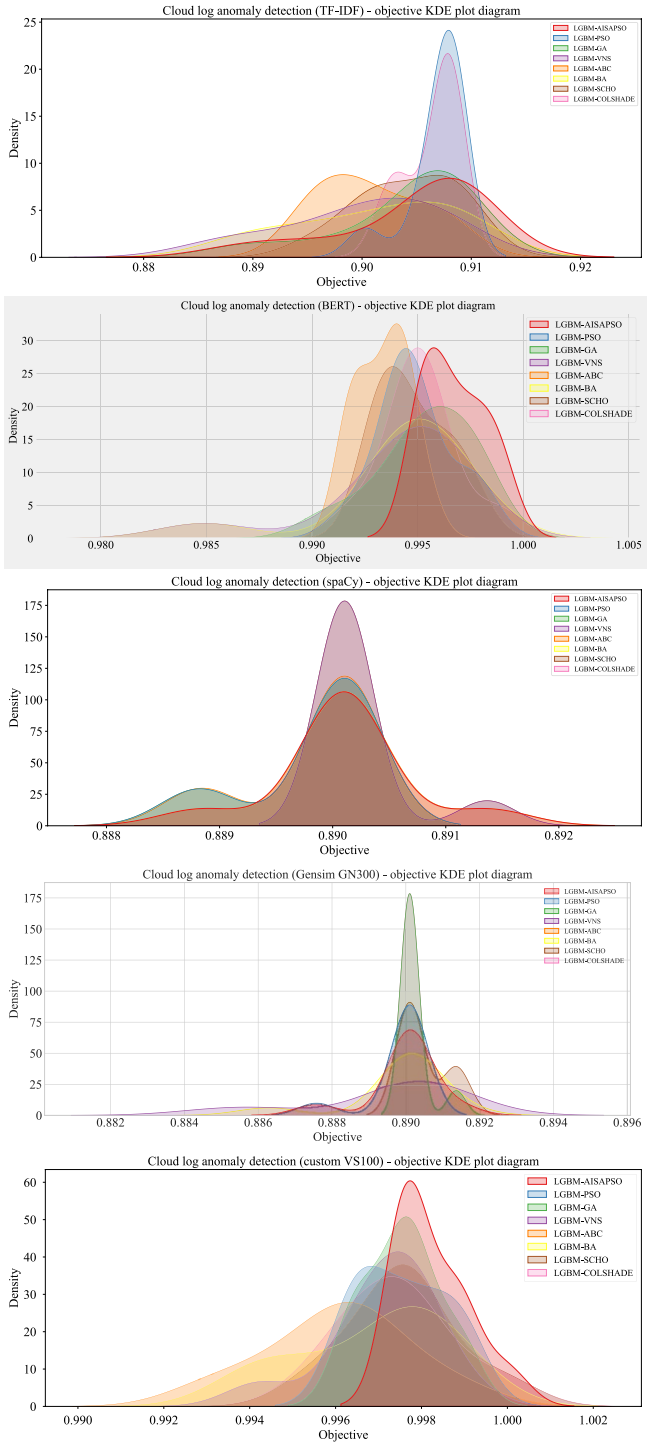


Fig. 7. KDE graphs for the fitness for TF-IDF (top), BERT, spaCy, Gensim Google News and Gensim custom (bottom) simulations.

Best model interpretation

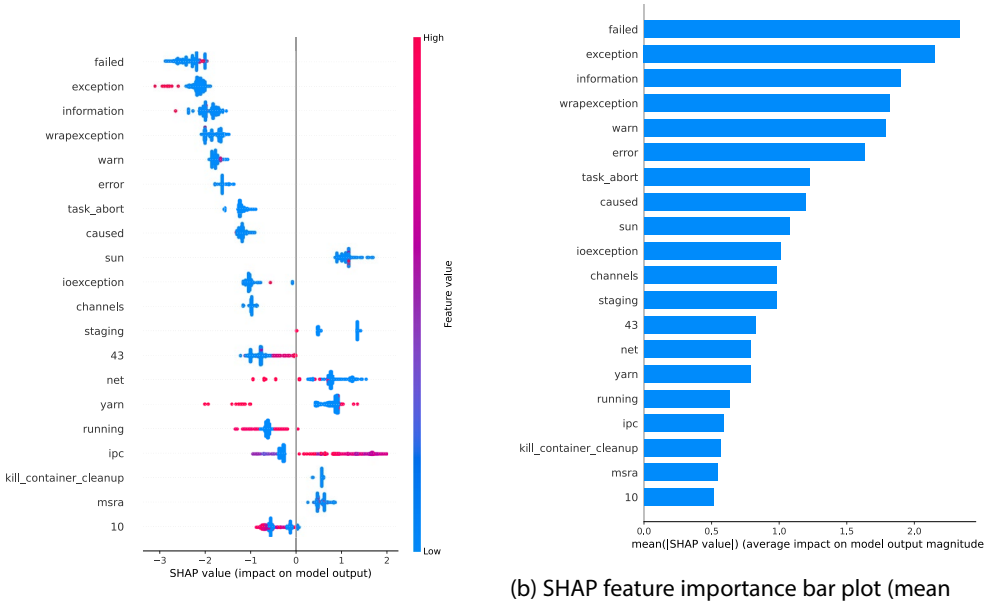
The value of classification models cannot be assessed by accuracy metrics alone. Although predictive precision is essential for practical deployment, accuracy on its own offers only a narrow view of performance. Equally critical is understanding the rationale behind a model's predictions. Interpretability fulfills several roles: it exposes hidden biases, identifies which features exert the greatest influence, and reinforces the integrity of the analytical pipeline. It can also reveal upstream weaknesses, such as flaws in data acquisition or preprocessing. In fields like NLP and computer vision, insight into a model's decision-making not only improves feature engineering but also enhances trust in real-world use. Yet transparency is notoriously difficult to achieve. As modern architectures grow deeper and more complex, their internal logic becomes increasingly opaque, complicating efforts to

LightGBM	PSO	GA	VNS	ABC	BA	SCHO	COLSHADE
TF-IDF	.042	.034	.026	.028	.032	.037	.041
BERT	.039	.035	.027	.025	.031	.033	.037
spaCy	.038	.037	.046	.038	.044	.045	.044
Gensim Google News (300)	.039	.045	.039	.041	.039	**.052**	.045
Gensim custom (100)	.039	.036	.035	.026	.029	.039	.037

Table 23. Wilcoxon signed-rank test scores (AISAPSO vs others) for each of three experiments.

Approach	Metric	Normal	Anomalous	Accuracy	Macro ave	Weight ave
TF-IDF LGBM-AISAPSO	Precision	.999067	.851103	.985429	.925085	.987334
	Recall	.985094	.989316	.985429	.987205	.985429
	f1-score	.992031	.915020	.985429	.953525	.985925
BERT LGBM-AISAPSO	Precision	1	.999816	.999831	.999908	.999831
	Recall	.997863	1	.999831	.998932	.999831
	f1-score	.998930	.999908	.999831	.999419	.999830
spaCy LGBM-AISAPSO	Precision	.983707	1	.984751	.991854	.985000
	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
Gensim Google News (300)	Precision	.983707	1	.984751	.991854	.984999
LGBM-AISAPSO	Recall	1	.807692	.984751	.903846	.984751
	f1-score	.991787	.893617	.984751	.942702	.984002
Gensim custom trained (100)	Precision	1	1	1	1	1
LGBM-AISAPSO	Recall	1	1	1	1	1
	f1-score	1	1	1	1	1
AdaBoost	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
CatBoost	Precision	.983529	1	.984581	.991765	.984835
	recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
CNN	Precision	.983529	1	.984581	.991765	.984835
	recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
Decision Tree	Precision	.983520	.992105	.984073	.987813	.984201
	recall	.999448	.805556	.984073	.902502	.984073
	f1-score	.991420	.889151	.984073	.940286	.983311
KNN	Precision	.983699	.992126	.984243	.987912	.984367
	recall	.999448	.807692	.984243	.903570	.984243
	f1-score	.991511	.890459	.984243	.940985	.983498
LightGBM	Precision	.983529	1	.984581	.991765	.984835
	recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
MLP	Precision	.983174	1	.984243	.991587	.984508
	recall	1	.801282	.984243	.900641	.984243
	f1-score	.991515	.889680	.984243	.940598	.983440
Random forest	Precision	.983351	1	.984412	.991676	.984672
	Recall	1	.803419	.984412	.901709	.984412
	f1-score	.991606	.890995	.984412	.941301	.983628
XGBoost	Precision	.983529	1	.984581	.991765	.984835
	Recall	1	.805556	.984581	.902778	.984581
	f1-score	.991696	.892308	.984581	.942002	.983815
	Support	5434	468			

Table 24. Comparisons of best-synthesized models against baseline models—detailed metrics.



(a) SHAP summary (beeswarm) plot

(b) SHAP feature importance bar plot (mean absolute SHAP values)

Fig. 8. SHAP summary and bar plots.

validate whether predictions align with human reasoning, to detect systematic errors, or to fully understand the mechanisms driving outcomes.

To address these challenges, this study employs SHAP (SHapley Additive exPlanations)[14], a unified, theoretically grounded method for interpreting model behavior. SHAP decomposes predictions into additive contributions from individual input features, clarifying how each variable shapes the output. In this work, SHAP was applied to the best-performing configuration, LGBM-AISAPSO combined with custom trained Gensim (vector size 100) preprocessing, and the resulting interpretability visualization is shown in Fig. 8.

The left-hand panel presents a SHAP summary (beeswarm) visualization, capturing how individual tokens contribute to the NLP classifier's decisions. Each dot represents a feature–instance pair, positioned according to its SHAP value and colored by the magnitude of the underlying feature (like token frequency or embedding weight). Tokens ranked higher on the vertical axis exert stronger influence on the model's predictions. As expected, terms like "failed", "exception", "information", "warn", and "error" show pronounced positive contributions, highlighting their central role in distinguishing anomalous from normal log events. This fine-grained view reveals how particular words systematically push outputs toward specific categories, exposing the linguistic cues the model relies on.

The right-hand panel complements this with a SHAP bar chart that aggregates mean absolute SHAP values across the dataset. Unlike the beeswarm plot, it removes directional effects and instead ranks features by their overall explanatory strength. Here again, tokens like "failed", "exception", "information", "wrapexception", "warn", and "error" emerge as the dominant drivers of model behavior. This global perspective is especially valuable for prioritizing features, guiding interpretability analysis, and refining monitoring or moderation strategies by flagging the terms most responsible for shaping predictions.

Taken together, the two plots provide complementary insights: the beeswarm plot uncovers token-level variability across individual predictions, while the bar chart highlights corpus-wide trends. Their combination improves transparency, strengthens interpretability, and builds confidence in deploying NLP classifiers for sensitive anomaly detection tasks.

Limitations of the study

Although the proposed framework demonstrated strong performance across multiple experimental settings, several limitations should be acknowledged.

First, the evaluation was conducted on a single publicly available dataset. While representative, this dataset may exhibit structural characteristics that facilitate high classification performance. Consequently, the generalizability of the proposed approach to other types of log data, particularly those with different formats, noise levels, or evolving patterns, requires further investigation.

Second, the hyperparameter optimization process introduces additional computational overhead during the training phase. Although this cost is incurred offline and does not affect real-time inference, it may become significant for very large datasets or more complex model configurations.

Third, baseline models were evaluated using standard parameter configurations rather than undergoing the same optimization procedure. While this design enables clearer isolation of the impact of the proposed optimization strategy, it may limit the completeness of the comparative analysis.

Fourth, although interpretability was enhanced through SHAP analysis, further investigation into model transparency and explainability in real-world deployment scenarios remains an important direction.

Fifth, differences in dataset size and embedding dimensionality across preprocessing methods may influence the direct comparability of results, as configurations were intentionally adapted to reflect practical and computational constraints.

Finally, although the framework was designed with deployment constraints in mind, including reduced feature dimensionality and bounded model complexity, empirical validation on embedded hardware platforms was not conducted.

Future work will focus on extending the evaluation to more diverse datasets, including real-time and streaming log environments, as well as investigating alternative model architectures and optimization strategies. In addition, further research will explore reducing optimization cost through surrogate modeling or adaptive search techniques, and will include empirical validation on embedded platforms such as ESP32, with emphasis on memory usage, inference latency, and energy efficiency.

Conclusion

The rapid proliferation of cloud computing has fundamentally reshaped the way organizations store, process, and protect their digital assets. However, this transition has also intensified the challenges of overseeing such environments, where massive streams of log data must be continuously parsed to identify anomalies, configuration errors, or emerging security risks. Artificial intelligence offers a powerful means of automating this monitoring, promising improvements in efficiency, resilience, and threat detection. Yet, its integration is not without complications. Alongside its benefits, AI introduces pressing concerns around interpretability, accountability, algorithmic bias, and the governance of automated decisions. Meeting these challenges demands cross-disciplinary approaches that balance technological capability with organizational priorities and regulatory compliance.

This work introduces a unified framework designed to detect and classify high-risk log patterns, with the overarching goal of enhancing anomaly detection accuracy in large-scale cloud systems. The study systematically evaluated several advanced NLP representations, including TF-IDF, BERT, spaCy Word2Vec, Gensim trained on Google News embeddings, and a custom-trained Gensim model, each paired with a LightGBM classifier optimized through metaheuristic search. Experimental findings demonstrate that the framework delivers strong capability in identifying and categorizing anomalous or security-sensitive events, consequently enabling automated log surveillance and reinforcing cloud reliability and security. A newly developed PSO variant was embedded into the optimization process, further boosting classifier performance. Of all the tested pipelines, the configuration combining custom trained Gensim embeddings (vector size 100) with LightGBM tuned via the proposed AISAPSO optimizer produced the most effective results, achieving a perfect accuracy of 100% in this evaluation. It is also important to note that lightweight structures were examined (particularly custom trained Gensim and LightGBM configurations), to support possible practical deployment on different microcontrollers that have limited resources.

The combination of advanced NLP-based preprocessing, LightGBM classification, and AISAPSO-driven metaheuristic optimization has important consequences for both cloud security management and system governance. From a practical standpoint, the proposed optimization framework is intended primarily for offline model preparation, while deployment relies on the resulting optimized classifier with low inference overhead. By achieving high accuracy while remaining scalable, the proposed framework equips organizations with a powerful mechanism for proactive surveillance of large-scale log data. For policymakers and industry stakeholders, such data-centric approaches can serve as practical tools to strengthen regulatory compliance, reduce security risks, and improve the reliability of cloud environments. Equally important, the integration of explainable AI techniques such as SHAP analysis enhances transparency and accountability, directly addressing the growing demand for interpretable AI in sensitive domains like cloud security monitoring.

Nevertheless, several constraints also affected this study. Limited access to large and up-to-date cloud log datasets restricted opportunities for ongoing validation and stress testing. Moreover, the computational intensity of the optimization procedures imposed boundaries on the number of algorithms, parameter configurations, population sizes, and iteration counts that could be realistically evaluated. Looking forward, future work will seek to scale the framework as computational resources grow, explore alternative preprocessing methods, and adapt the proposed optimization strategy to a broader range of security and infrastructure management challenges in cloud computing.

Data availability

The study employs an openly available dataset, accessible via Kaggle at https://www.kaggle.com/datasets/krishd123/log-data-for-anomaly-detection.

Received: 2 December 2025; Accepted: 30 April 2026
Published online: 13 May 2026

References

1. Chen, Z., Liu, J., Gu, W., Su, Y. & Lyu, M.R. Experience report: Deep learning-based system log analysis for anomaly detection. arXiv preprint arXiv:2107.05908 (2021).
2. Yuan, Y., Adhatarao, S.S., Lin, M., Yuan, Y., Liu, Z. & Fu, X. Ada: Adaptive deep log anomaly detector. In *IEEE Infocom 2020-IEEE Conference on Computer Communications*, 2449–2458 (IEEE, 2020).
3. Jiang, Z., Liu, J., Chen, Z., Li, Y., Huang, J., Huo, Y., He, P., Gu, J. & Lyu, M.R. Lilac: Log parsing using llms with adaptive parsing cache. In *Proceedings of the ACM on Software Engineering 1(FSE)*, 137–160 (2024).

4. Landauer, M., Onder, S., Skopik, F. & Wurzenberger, M. Deep learning for anomaly detection in log data: A survey. *Mach. Learn. Appl.* **12**, 100470 (2023).
5. Mladenovic, D. et al. Sentiment classification for insider threat identification using metaheuristic optimized machine learning classifiers. *Sci. Rep.* **14**(1), 25731 (2024).
6. Jaiswal, G., Rani, R., Mangotra, H. & Sharma, A. Integration of hyperspectral imaging and autoencoders: Benefits, applications, hyperparameter tunning and challenges. *Comput. Sci. Rev.* **50**, 100584 (2023).
7. Purnomo, H. et al. Metaheuristics approach for hyperparameter tuning of convolutional neural network. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* **8**(3), 340–345 (2024).
8. Christian, H., Agus, M.P. & Suhartono, D. Single document automatic text summarization using term frequency-inverse document frequency (tf-idf). *ComTech Comput. Math. Eng. Appl.* **7**(4), 285–294 (2016).
9. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics* (2019).
10. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013).
11. Kuo, C. *The Handbook of NLP with Gensim: Leverage Topic Modeling to Uncover Hidden Patterns, Themes, and Valuable Insights Within Textual Data* (Packt Publishing Ltd, 2023).
12. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.* **30** (2017)
13. Kennedy, J.& Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95—International Conference on Neural Networks*, vol. 4, 1942–19484 (1995).
14. Lundberg, S.M. & Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17*, 4768–4777 (Curran Associates Inc., 2017).
15. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z. & Tang, J. Gpt understands, too. (AI Open, 2023)
16. Kumar, A., Sharma, R. & Bedi, P. Towards optimal NLP solutions: Analyzing GPT and llama-2 models across model scale, dataset size, and task diversity. *Eng. Technol. Appl. Sci. Res.* **14**(3), 14219–14224 (2024).
17. Yang, L. & Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **415**, 295–316 (2020).
18. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**(1), 67–82 (1997).
19. Saheed, Y. K. & Chukwuere, J. E. Xai-enhanced adversarial resilient deep learning framework for transparent and secure edge deployment in consumer internet of things/industrial internet of things environments. *Comput. Electr. Eng.* **133**, 111080 (2026).
20. Saheed, Y.K. & Chukwuere, J.E. Cps-iiot-p2attention: Explainable privacy-preserving with scaled dot-product attention in cyber physical system-industrial IOT network. *IEEE Access*. (2025)
21. Saheed, Y.K. & Chukwuere, J.E. Autonomous llm agent: A memory-augmented, edge-optimized shap explanations with zero-day attack resilience in IOT/industrial IOT networks. *IEEE Internet Things J.* (2025).
22. Saheed, Y. K., Abdulganiyu, O. H. & Ait Tchakoucht, T. Modified genetic algorithm and fine-tuned long short-term memory network for intrusion detection in the internet of things networks with edge capabilities. *Appl. Soft Comput.* **155**, 111434 (2024).
23. Saheed, Y. K. & Chukwuere, J. E. Xaiensembletl-iov: A new explainable artificial intelligence ensemble transfer learning for zero-day botnet attack detection in the internet of vehicles. *Results Eng.* **24**, 103171 (2024).
24. Adeyiola, A.Q., Saheed, Y.K., Misra, S. & Chockalingam, S. Metaheuristic firefly and c5. 0 algorithms based intrusion detection for critical infrastructures. In *2023 3rd International Conference on Applied Artificial Intelligence (ICAPAI)*, 1–7 (IEEE, 2023).
25. Mirjalili, S. & Mirjalili, S. Genetic algorithm. Evolutionary algorithms and neural networks: Theory and applications, 43–55 (2019).
26. Karaboga, D. & Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **39**(3), 459–471 (2007).
27. Mladenović, N. & Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997).
28. Mirjalili, S. et al. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **114**, 163–191 (2017).
29. Mirjalili, S. & Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67 (2016).
30. Yang, X.-S. & He, X. Bat algorithm: Literature review and applications. *Int. J. Bio-Inspir. Comput.* **5**(3), 141–149 (2013) (**PMID: 55093**).
31. Shi, Y. An optimization algorithm based on brainstorming process. In *Emerging Research on Swarm Intelligence and Algorithm Optimization*, 1–35 (IGI Global, 2015).
32. Abualigah, L., Elaziz, M. A., Sumari, P., Geem, Z. W. & Gandomi, A. H. Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **191**, 116158 (2022).
33. Khishe, M. & Mosavi, M. R. Chimp optimization algorithm. *Expert Syst. Appl.* **149**, 113338 (2020).
34. Bai, J. et al. A sinh cosh optimizer. *Knowl.-Based Syst.* **282**, 111081 (2023).
35. Gurrola-Ramos, J., Hernàndez-Aguirre, A. & Dalmau-Cedeño, O. Colshade for real-world single-objective constrained optimization problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, 1–8 (2020).
36. Cuk, A. et al. Tuning attention based long-short term memory neural networks for Parkinson's disease detection using modified metaheuristics. *Sci. Rep.* **14**(1), 4309 (2024).
37. Zivkovic, M., Bacanin, N., Zivkovic, T., Jovanovic, L., Kaljevic, J. & Antonijevic, M. Parkinson's detection from gait time series classification using LSTM tuned by modified RSA algorithm. In *International Conference on Communication and Computational Technologies*, 119–134 (Springer, 2024).
38. Zivkovic, M., Antonijevic, M., Jovanovic, L., Krasic, M., Bacanin, N., Zivkovic, T. & Milovanovic, M. Ocular disease diagnosis using cnns optimized by modified variable neighborhood search algorithm. In *International Joint Conference on Advances in Computational Intelligence*, 99–112 (Springer, 2025).
39. Todorovic, M. et al. Improving audit opinion prediction accuracy using metaheuristics-tuned xgboost algorithm with interpretable results through shap value analysis. *Appl. Soft Comput.* **149**, 110955 (2023).
40. Villoth, J. P. et al. Two-tier deep and machine learning approach optimized by adaptive multi-population firefly algorithm for software defects prediction. *Neurocomputing* **630**, 129695 (2025).
41. Dobrojevic, M., Jovanovic, L., Babic, L., Cajic, M., Zivkovic, T., Zivkovic, M., Muthusamy, S., Antonijevic, M. & Bacanin, N. Cyberbullying sexism harassment identification by metaheurustics-tuned extreme gradient boosting. *Comput. Mater. Continua* **80**(3) (2024).
42. Bacanin, N. et al. The explainable potential of coupling hybridized metaheuristics, xgboost, and shap in revealing toluene behavior in the atmosphere. *Sci. Total Environ.* **929**, 172195 (2024).
43. Tasic, A. et al. Towards sustainable societies: Convolutional neural networks optimized by modified crayfish optimization algorithm aided by adaboost and xgboost for waste classification tasks. *Appl. Soft Comput.* **175**, 113086 (2025).
44. Dakic, P. et al. Intrusion detection using metaheuristic optimization within IOT/IIOT systems and software of autonomous vehicles. *Sci. Rep.* **14**(1), 22884 (2024).
45. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. & Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **26** (2013)

46. Altinok, D. *Mastering spaCy: An End-to-end Practical Guide to Implementing NLP Applications Using the Python Ecosystem* (Packt Publishing Ltd, 2021).
47. Jugran, S., Kumar, A., Tyagi, B.S. & Anand, V. Extractive automatic text summarization using spacy in python & nlp. In *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 582–585 (IEEE, 2021).
48. Antonijevic, M. et al. Intrusion detection in metaverse environment internet of things systems by metaheuristics tuned two level framework. *Sci. Rep.* **15**(1), 3555 (2025).
49. Li, L. et al. A lightgbm-based strategy to predict tunnel rockmass class from TBM construction data for building control. *Adv. Eng. Inform.* **58**, 102130 (2023).
50. Wang, D.-N., Li, L. & Zhao, D. Corporate finance risk prediction based on lightgbm. *Inf. Sci.* **602**, 259–268 (2022).
51. Lao, Z. et al. Intelligent fault diagnosis for rail transit switch machine based on adaptive feature selection and improved lightgbm. *Eng. Fail. Anal.* **148**, 107219 (2023).
52. Salb, M. et al. Cloud spot instance price forecasting multi-headed models tuned using modified PSO. *J. King Saud Univ.-Sci.* **36**(11), 103473 (2024).
53. Jovanovic, L., Zivkovic, M., Zivkovic, T., Bacanin, N., Dubljanin, D. & Malisic, S. Steam players count prediction using GRU optimized by modified PSO. In *2024 32nd Telecommunications Forum (TELFOR)*, 1–4 (IEEE, 2024).
54. Jovanovic, L., Bacanin, N., Petrovic, A., Zivkovic, M., Antonijevic, M., Gajic, V., Elsayed, M.M. & Abouhawwash, M. Exploring artificial intelligence potential in solar energy production forecasting: Methodology based on modified PSO optimized attention augmented recurrent networks. *Sustain. Comput. Inform. Syst.* 101174 (2025).
55. Luo, W., Lin, X., Li, C., Yang, S. & Shi, Y. Benchmark functions for CEC 2022 competition on seeking multiple optima in dynamic environments (2022). https://arxiv.org/abs/2201.00523
56. Rahnamayan, S., Tizhoosh, H.R. & Salama, M.M. Quasi-oppositional differential evolution. In *2007 IEEE Congress on Evolutionary Computation*, 2229–2236 (IEEE, 2007).
57. Dubey, K. Log Data for Anomaly Detection. Database Contents License (DbCL) v1.0 (2022). https://www.kaggle.com/datasets/krishd123/log-data-for-anomaly-detection
58. Hossin, M. & Sulaiman, M. N. A review on evaluation metrics for data classification evaluations. *Int. J. Data Mining Knowl. Manag. Process* **5**(2), 1 (2015).
59. Matthews, B.W. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochim. Biophys. Acta BBA-Protein Struct.* **405**(2), 442–451 (1975).
60. LaTorre, A. et al. A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. *Swarm Evol. Comput.* **67**, 100973 (2021).
61. Schultz, B. B. Levene's test for relative variation. *Syst. Biol.* **34**(4), 449–456. https://doi.org/10.1093/sysbio/34.4.449 (1985).
62. Shapiro, S.S. & Francia, R.S. An approximate analysis of variance test for normality. *J. Am. Stat. Assoc.* 67(337), 215–216. https://doi.org/10.1080/01621459.1972.10481232 (1972).
63. Woolson, R. F. *Wilcoxon signed-rank test* https://doi.org/10.1002/0470011815.b2a15177 (2005).
64. Hastie, T., Rosset, S., Zhu, J. & Zou, H. Multi-class adaboost. Statistics and its. *Interface* **2**(3), 349–360 (2009).
65. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V. & Gulin, A. Catboost: Unbiased boosting with categorical features. *Adv. Neural Inf. Process. Syst.* **31** (2018).
66. Gu, J. et al. Recent advances in convolutional neural networks. *Pattern Recogn.* **77**, 354–377 (2018).
67. Ville, B. Decision trees. *WIREs Comput. Stat.* **5**(6), 448–455. https://doi.org/10.1002/wics.1278 (2013).
68. Kramer, O. *K-Nearest Neighbors*, 13–23. https://doi.org/10.1007/978-3-642-38652-7_2 (Springer, 2013).
69. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
70. Chen, T. & Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794 (2016).

Acknowledgements

This work was supported by the Science Fund of the Republic of Serbia, Grant No. 7373, Characterizing crises-caused air pollution alternations using an artificial intelligence-based framework, crAIRsis and Grant No. 7502, Intelligent Multi-Agent Control and Optimization applied to Green Buildings and Environmental Monitoring Drone Swarms, ECOSwarm.

Author contributions

S.D., S.S. and B.N. - writing original draft; S.A., T.Z. and V.S. - validation and visualization; M.Z. and N.B. - writing review and editing, simulations, data acquisition.

Funding

No funding was received to support this study.

Declarations

Competing interests

The authors declare no competing interests.

Use of AI tools

Artificial intelligence–based tools were used in this paper solely to assist with stylistic and grammatical refinement, including paraphrasing for clarity, correcting language errors, and improving sentence flow. After using the tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article. All scientific content, analyses, interpretations, and conclusions are entirely the authors' original work.

Additional information

Correspondence and requests for materials should be addressed to N.B.