THE JOURNAL OF APPLIED ENGINEERING AND TECHNOLOGIES
2026 Vol. 1, No. 3, ISSN (O): 3154-7877

Resilience-Oriented Cyber Risk Recognition in Smart Grid Communication Infrastructures

Ka Ho Chan[1], Tsz Yan Wong[1], Hoi Lam Lee[1*]
[1]Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong SAR, China
Corresponding Author*: Hoi Lam Lee E-mail: hoilam.lee@x.hku.hk

ARTICLE INFO

Keywords
Cloud-native security
microservice traffic
anomaly detection
ensemble learning
explainable AI
SHAP
Kubernetes security
service-level features

Published:
01 September 2026

ABSTRACT

Cloud-native applications rely on large-scale microservice communication, where abnormal service-to-service traffic may indicate lateral movement, API abuse, misconfiguration, or compromised containers. Existing intrusion detection approaches are often designed for traditional network environments and may not capture the dynamic traffic behavior of containerized systems. This study proposes an interpretable ensemble learning model for anomaly detection in cloud-native microservice traffic. The framework combines XGBoost, Extra Trees, and Gradient Boosting classifiers to detect abnormal service communication patterns, while SHAP analysis is used to interpret service-level feature contributions. Experiments are performed on a Kubernetes-based testbed containing 126 microservices and 3.4 million service flow records generated under normal workloads and attack scenarios, including API flooding, privilege escalation attempts, abnormal east-west traffic, and container scanning. After feature extraction, 39 service-level features are retained, including request frequency, response latency variance, failed request ratio, inter-service connection density, port diversity, and abnormal destination concentration. The proposed model achieves 98.05% accuracy, 97.22% F1-score, and 98.73% AUC in binary anomaly detection. For multi-class attack identification, it obtains a macro-F1 of 95.84%, outperforming a CNN-LSTM baseline by 2.63%. SHAP interpretation shows that failed request ratio, sudden increases in east-west traffic, service call entropy, and response-time instability are the most important indicators of microservice anomalies. The findings demonstrate that explainable ensemble learning can improve both detection performance and operational interpretability in cloud-native security monitoring.

1. Introduction

Cloud-native applications have become a core architecture for large-scale online services. By decomposing a monolithic application into many small and independently deployable services, cloud-native systems improve release speed, fault isolation, resource utilization, and elastic scaling. Kubernetes further strengthens these advantages by supporting container orchestration, service discovery, load balancing, and automated deployment. However, the same architectural features also increase the difficulty of security monitoring. A single business request may pass through many services, and the communication path may change with autoscaling, rolling updates, service migration, or policy adjustment (Batchu, 2025; Yin et al., 2026). In this environment, abnormal

Citation：Chan, K. H., Wong, T. Y., & Lee, H. L. (2026). Resilience-oriented cyber risk recognition in smart grid communication infrastructures. *The Journal of Applied Engineering and Technologies*, 1(3), 36-43.

https://doi.org/10.64744/tjaet.2026.283.

service-to-service traffic may indicate lateral movement, API abuse, container scanning, privilege escalation, misconfiguration, or unstable service dependencies. Traditional intrusion detection methods often rely on fixed network boundaries, packet-level features, or known attack signatures, which are less effective in cloud-native systems because containers are short-lived, service IPs change frequently, and east-west traffic becomes more important than north-south traffic (Iavich et al., 2025; Yang, 2026).

Security monitoring in microservice systems increasingly depends on runtime telemetry, including logs, metrics, traces, service graphs, and network flow records. These data sources provide useful information about request status, service dependency, traffic direction, response latency, and error propagation. Recent studies have used such telemetry to detect microservice anomalies and cloud-native security risks (Deng & Yang, 2026). In particular, interpretable ensemble learning with SHAP-based explanation has shown value in network traffic anomaly detection because it can combine efficient structured-feature learning with feature-level alarm interpretation (Shao, 2026). This idea is relevant to cloud-native traffic security, where engineers not only need to know that an anomaly exists, but also need to understand whether the alarm is related to failed requests, sudden east-west traffic growth, abnormal service call order, unstable response time, or irregular traffic concentration (Gaddam, 2026; Raeiszadeh et al., 2026). Machine learning models have been widely applied to anomaly detection in microservice and containerized environments (Nguyen & Ilin, 2026; Zhang, 2026). Tree-based models such as Random Forest, XGBoost, Gradient Boosting, and Isolation Forest are suitable for structured traffic features and can provide relatively low computational cost (Chen et al., 2025; Ness et al., 2025). Deep learning models can capture temporal patterns, long-range dependency changes, and complex service interaction structures, and they have been used in trace-based anomaly detection, service dependency modeling, and runtime behavior analysis (Liang et al., 2025; Pokkuluri & Khang, 2025). However, these models often require large training datasets, high computing resources, and careful parameter tuning. Their alarm outputs may also be difficult for platform engineers to interpret in real operation scenarios (Huang, 2026). In cloud-native systems, this limitation is important because an unexplained alarm may not support fast incident response, root-cause analysis, or security policy adjustment (Ma, 2026; Mittal, 2026). Existing studies still leave several gaps. Many approaches focus on host-level events, resource metrics, or application logs, while service-level flow features receive less attention. Some methods detect abnormal behavior but do not clearly explain which traffic characteristics contribute to the detection result (Jency & Ramar, 2025; Li & Liu, 2026). Others are designed for general network intrusion detection and do not fully consider the dynamic service topology, high-frequency service calls, and unstable traffic paths in Kubernetes environments (Gao et al., 2026; Okafor et al., 2025). As a result, there remains a need for a lightweight, interpretable, and service-level anomaly detection method that can work with cloud-native traffic data and provide practical explanations for security engineers (Shonubi & Adelere, 2025; Wang et al., 2025).

To address these gaps, this study proposes an interpretable ensemble learning model for cloud-native microservice traffic anomaly detection. The model combines XGBoost, Extra Trees, and Gradient Boosting to improve detection robustness across different traffic patterns. SHAP is used to explain the contribution of service-level flow features, including request volume, response status, latency, byte transfer, traffic direction, and service interaction characteristics. Experiments are conducted on a Kubernetes-based testbed containing 126 microservices and 3.4 million service flow records collected under normal workloads and attack scenarios. The purpose of this study is

to improve anomaly detection accuracy while making the detection results understandable and actionable. The significance of the study lies in providing a practical security monitoring approach for cloud-native systems, helping engineers identify abnormal service behavior, interpret alarm causes, and strengthen runtime protection in dynamic Kubernetes environments.

2. Materials and Methods

2.1 Samples and Study Objects

The samples were collected from a Kubernetes-based cloud-native testbed. The testbed contained 126 microservices deployed in container pods and service nodes. A total of 3.4 million service flow records were collected under normal workloads and controlled attack conditions. This study focused on service-to-service traffic inside a microservice system. Each record represented one service flow and included request behavior, response status, latency change, connection pattern, and destination distribution. Normal traffic came from regular API calls, user requests, service discovery, and background system tasks. Abnormal traffic included API flooding, privilege escalation attempts, abnormal east-west traffic, and container scanning. These samples reflected common security risks in Kubernetes systems, where services are often created, scaled, updated, and removed.

2.2 Experimental Design and Control Settings

The experiment tested whether ensemble learning could improve anomaly detection for cloud-native microservice traffic. The experimental group used an ensemble model that combined XGBoost, Extra Trees, and Gradient Boosting. These models were selected because they can process structured service-level features and learn nonlinear traffic patterns. The control groups included standalone XGBoost, Extra Trees, Gradient Boosting, Random Forest, Logistic Regression, and CNN-LSTM models. The CNN-LSTM model was used as a deep learning baseline because it is often applied to sequence-based traffic detection. Binary classification was used to separate normal and abnormal service flows. Multi-class classification was used to identify API flooding, privilege escalation attempts, abnormal east-west traffic, container scanning, and normal traffic. All models used the same feature set and the same training, validation, and test sets to keep the comparison fair.

2.3 Measurement Methods and Quality Control

Service-level flow features were extracted from Kubernetes traffic logs, service mesh records, request traces, and container network metadata. A total of 39 features were retained after feature screening. The measured variables included request frequency, response latency variance, failed request ratio, inter-service connection density, port diversity, abnormal destination concentration, service call entropy, response code distribution, and east-west traffic growth rate. These variables were used because microservice attacks often change service communication behavior before clear host-level signals appear. During quality control, duplicate records, incomplete flows, missing labels, invalid timestamps, and non-numeric errors were removed. Time windows with system restarts or planned deployment changes were excluded to reduce noise. Continuous variables were scaled before training. Highly repeated and weak variables were removed. The test set was kept separate from training and parameter tuning to prevent data leakage.

2.4 Data Processing and Model Formulas

All service flow records were cleaned, scaled, and divided into training, validation, and test sets by stratified sampling. The ensemble model generated prediction probabilities from XGBoost,

Extra Trees, and Gradient Boosting. The final class was determined by the average probability output of the three models. The prediction rule was defined as:

$$\hat{y} = \arg\max_{c} \left(\frac{1}{3} \sum_{m=1}^{3} P_m (y=c|x) \right)$$

where is the service-level feature vector, is the traffic class, and is the probability produced by the mmm-th model. Model performance was measured using accuracy, precision, recall, F1-score, macro-F1, AUC, and false positive rate. The macro-F1 score was calculated as:

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^{K} F1_k$$

where is the number of traffic classes, and is the F1-score of the -th class. Macro-F1 was used because it gives equal weight to each attack type and is suitable for uneven class distributions.

2.5 Model Interpretation and Evaluation Procedure

SHAP analysis was used to explain how each service-level feature affected the detection result. Global SHAP values were used to rank important features in the test set. Local SHAP values were used to explain single abnormal service flows. The interpretation focused on failed request ratio, sudden growth in east-west traffic, service call entropy, response-time instability, port diversity, and abnormal destination concentration. The proposed model was evaluated from four aspects: binary anomaly detection, multi-class attack identification, comparison with baseline models, and feature-level explanation. The results were compared with all control models under the same data partition and feature setting. This process tested detection accuracy, class-level stability, and operational interpretability in Kubernetes-based microservice security monitoring.

3. Results and Discussion

3.1 Binary Anomaly Detection Performance

The ensemble model achieved 98.05% accuracy, 97.22% F1-score, and 98.73% AUC in binary anomaly detection. These results show that service-level flow features can separate normal microservice traffic from abnormal service calls. The model performed better than single classifiers because the three base models learned different traffic patterns. XGBoost captured nonlinear relations among failed requests, latency changes, and traffic bursts. Extra Trees reduced unstable results through randomized tree splitting. Gradient Boosting improved the decision boundary by correcting previous errors. Compared with container security methods that rely mainly on host behavior or system logs, this method focuses on service-to-service communication. This design is more suitable for Kubernetes systems because abnormal east-west traffic may appear before clear host-level signs (Carcagni, 2025; Gong et al., 2025).

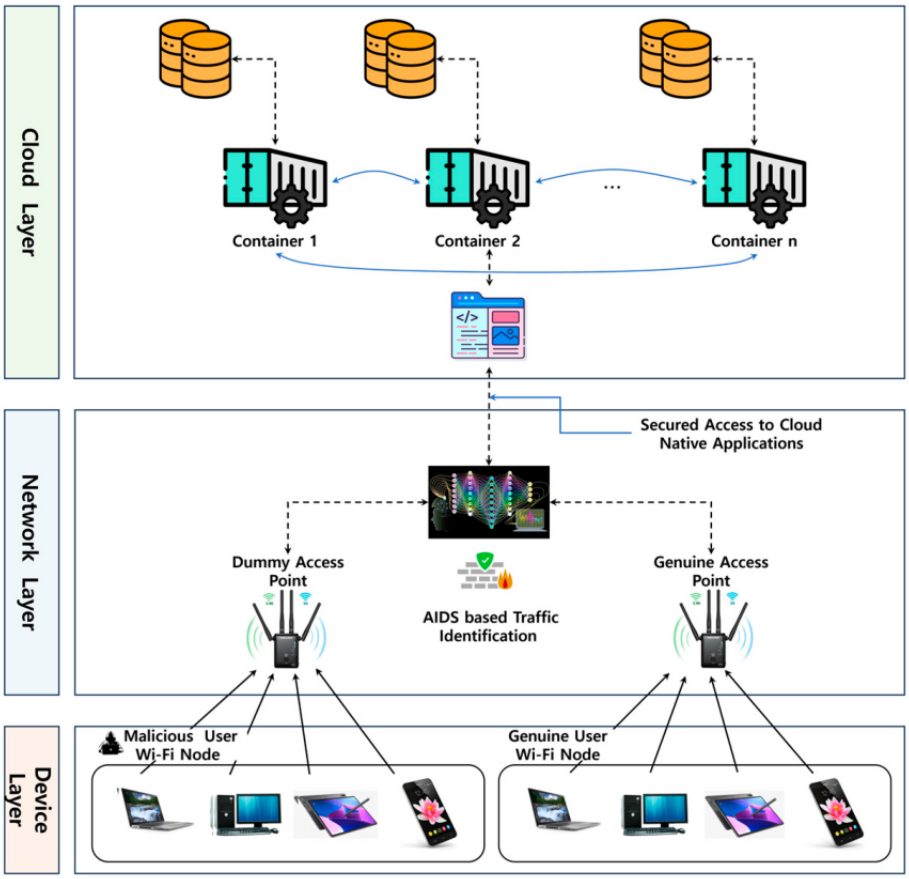

Fig. 1. Cloud-native microservice traffic anomaly detection framework.

3.2 Multi-Class Attack Identification

In the multi-class task, the ensemble model reached a macro-F1 score of 95.84% across API flooding, privilege escalation attempts, abnormal east-west traffic, container scanning, and normal traffic. The model performed well on API flooding because this attack caused a sharp increase in request frequency and failed request ratio. It also detected abnormal east-west traffic well because this attack changed inter-service connection density and destination concentration. Container scanning was mainly reflected by port diversity, repeated short connections, and unusual service access patterns (Ruambo et al., 2025; Zhang et al., 2026). Compared with the CNN-LSTM baseline, the model improved macro-F1 by 2.63%. This result suggests that structured service-flow features can be more useful than sequence-only features when attacks are reflected in service calls, response status, and connection distribution.

3.3 Feature Contribution and Interpretation

SHAP analysis showed that failed request ratio, sudden growth in east-west traffic, service call entropy, response-time instability, abnormal destination concentration, and port diversity were the main factors affecting anomaly detection. These findings match common cloud-native attack behavior. A high failed request ratio may appear during API abuse or privilege escalation attempts. A sudden increase in east-west traffic may indicate lateral movement inside the cluster. High service call entropy suggests that one service is communicating with many unusual destinations. Response-time instability may reflect repeated retries, overloaded services, or hidden attack traffic. Compared with black-box deep learning models, the ensemble model provides clearer feature-level evidence. This helps DevOps and security teams review alerts and connect model outputs with real service events (Nagasundari et al., 2025; Qi & Qiao, 2026).

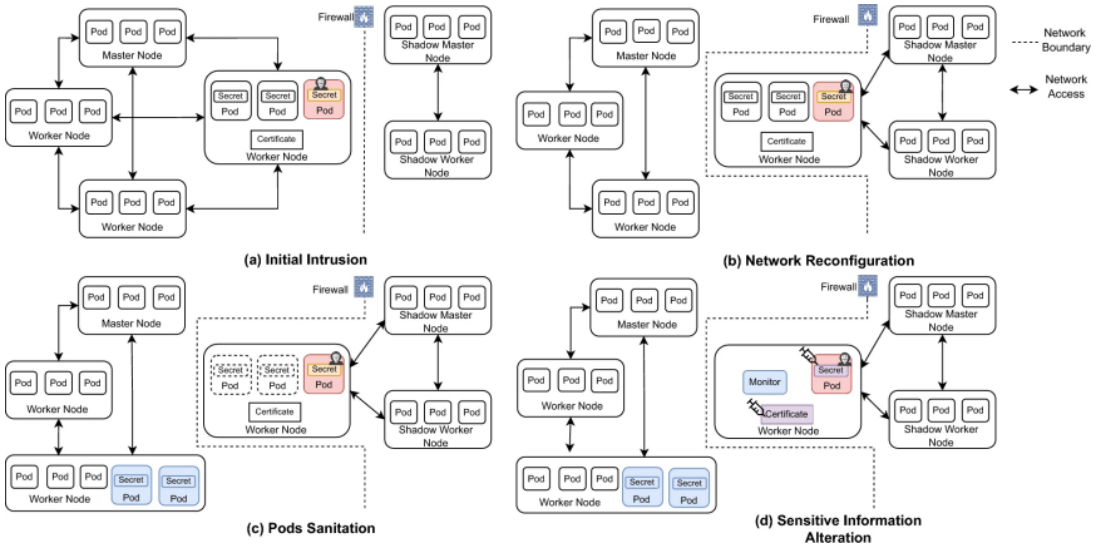

Fig. 2. Kubernetes behavior monitoring and anomaly response process.

3.4 Comparison with Existing Studies and Practical Value

Compared with earlier microservice and container anomaly detection studies, the proposed method has clearer value for cloud-native traffic monitoring. First, it uses service-level flow features rather than only host metrics, raw logs, or system calls. This helps detect abnormal service communication before host-level signs become obvious (Su & Zhang, n.d.). Second, the ensemble model achieved strong results without using a large neural network. This reduces training cost and makes deployment easier in real monitoring pipelines. Third, SHAP interpretation gives clear reasons for each detection result, which is useful for alert review, incident analysis, and later security auditing. This study still has limits. The experiments were conducted in a Kubernetes-based testbed, and the model should be tested in production clusters with more service types, changing workloads, and mixed attack strategies (Farid et al., 2025; Wang et al., 2026). Future work should examine online learning, service mesh integration, and lower-latency feature extraction for real-time cloud-native security monitoring.

4. Conclusion

This study proposed an interpretable ensemble learning method for anomaly detection in cloud-native microservice traffic. The model combined XGBoost, Extra Trees, and Gradient Boosting, and used service-level flow features to detect abnormal communication in Kubernetes systems. Experiments on a testbed with 126 microservices and 3.4 million service flow records showed strong results in both binary detection and multi-class attack identification. The model achieved high accuracy, F1-score, and AUC, and its macro-F1 was higher than that of the CNN-LSTM baseline. These results show that structured service-flow features can detect API flooding, privilege escalation attempts, abnormal east-west traffic, and container scanning. SHAP analysis showed that failed request ratio, sudden growth in east-west traffic, service call entropy, and response-time instability were the main signs of microservice anomalies. The main contribution of this study is that it combines ensemble learning with clear service-level explanation for cloud-native security monitoring. The method can support alert review, incident analysis, and security auditing in Kubernetes systems. This study still has some limits. The experiments were conducted in a controlled testbed, and the model has not been tested in production clusters with changing workloads, mixed service types, and complex attack strategies. Future work should test the method in real cloud environments, reduce feature extraction delay, and study its use with service mesh monitoring and online learning.

Acknowledgments
We are grateful to all respondents who participated in this study.

References

1. Batchu, K. C. (2025). Scalable Framework for Cloud Data Migration: Architecture, Strategy and Best Practices. *Journal Of Engineering And Computer Sciences, 4*(7), 604-611.
2. Carcagni, A. (2025). From Security Standard Definition to Centralized Posture Management: A Comprehensive Security Framework for Azure Kubernetes Service (Doctoral dissertation, Politecnico di Torino).
3. Chen, F., Liang, H., Yue, L., Xu, P., & Li, S. (2025). Low-Power Acceleration Architecture Design of Domestic Smart Chips for AI Loads.
4. Deng, X., & Yang, J. (2026). Design of a Quantitative Futures Trading Model Incorporating Multimodal Feature Fusion and Tail Risk Control. Available at SSRN 6702398.
5. Farid, M., Lim, H. S., Lee, C. P., Zarakovitis, C. C., & Chien, S. F. (2025). Optimizing Kubernetes with Multi-Objective Scheduling Algorithms: A 5G Perspective. *Computers, 14*(9), 390.
6. Gaddam, R. R. (2026). Cloud-Native Intelligent Computing Platforms for Secure, Scalable, and Automated Infrastructure. International Journal of AI, BigData, Computational and Management Studies, 118-128.
7. Gao, G., Gao, R., Gao, R., Zhou, H., & Lu, C. (2026, March). Performance Study of Enterprise SMS Communication Scheduling Mechanisms in Triple-Play Convergence Environments. In 2026 IEEE 8th International Conference on Communications, Information System and Computer Engineering (CISCE) (pp. 82-86). IEEE.
8. Gong, M., Deng, Y., Qi, N., Zou, Y., Xue, Z., & Zi, Y. (2025, August). Structure-learnable adapter fine-tuning for parameter-efficient large language models. In International Conference on Artificial Intelligence and Computational Engineering (AICE 2025) (Vol. 2025, pp. 225-230). IET.
9. Huang, R. (2026). ConfSched: Confidence-Threshold Routing for Efficient Edge-Cloud Activity Recognition. *World Journal of Engineering and Technology, 14*(2), 471-486.
10. Iavich, M., Svanadze, V., & Kovalchuk, O. (2025). A Deterministic, Rule-Based Framework for Detecting Anomalous IP Packet Fragmentation. *Future Internet, 18*(1), 19.
11. Jency, A., & Ramar, K. (2025). A review of abnormal behaviour detection in crowd for video surveillance: advances and trends, datasets, opportunities and prospects. *Expert Systems, 42*(4), e70013.
12. Li, Y., & Liu, S. (2026, May). A Study on Dynamic Optimization of Alerting Policies and Multi-Agent Decision-Making Mechanisms in Cloud Environments. In 2026 7th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT) (pp. 703-706). IEEE.
13. Liang, R., Fan, F., Liang, Y., & Li, S. (2025). Constructing an Adaptive Optimization Model for Ribbon Recommendation and Interface for User Habits.
14. Ma, Y. (2026). Industrial Financial Risk Prediction Model Based on Graph Neural Network and Knowledge Graph Inference. *Journal of Circuits, Systems and Computers, 35*(16), 2650103.
15. Mittal, A. (2026). AI-Driven Security Strategies in Cloud-Native Architectures: A Correlational Study of Adoption and Effectiveness (Doctoral dissertation, University of the Cumberlands).
16. Nagasundari, S., Manja, P., Mathur, P., & Honnavalli, P. B. (2025). Extensive review of threat models for DevSecOps. IEEE Access.
17. Ness, S., Eswarakrishnan, V., Sridharan, H., Shinde, V., Janapareddy, N. V. P., & Dhanawat, V.

(2025). Anomaly detection in network traffic using advanced machine learning techniques. *IEEE Access, 13*, 16133-16149.
18. Nguyen, M. H., & Ilin, D. (2026). Configurable Containerized Testbed for Dataset Building for Anomaly Detection in Data Processing Infrastructure using Machine Learning. *International Journal of Open Information Technologies, 14*(4), 75-84.
19. Okafor, K. C., Okafor, W. O., Longe, O. M., Ayogu, I. I., Anoh, K., & Adebisi, B. (2025). Scalable container-based time synchronization for smart grid data center networks. *Technologies, 13*(3), 105.
20. Pokkuluri, K. S., & Khang, A. (2025). Deep learning for identification of behavioral changes. In Behavioral Economics and Neuroeconomics of Health and Healthcare (pp. 65-78). IGI Global Scientific Publishing.
21. Qi, C., & Qiao, X. (2026). Efficient Data Sampling and Feature Selection Algorithms for Scalable Machine Learning Pipelines.
22. Raeiszadeh, M., Estrada-Solano, F., & Glitho, R. H. (2026). Anomalies Detection in Cloud-Native Architectures: Background, State-of-the-Art, and Research Directions. IEEE Communications Magazine.
23. Ruambo, F. A., Masanga, E. E., Lufyagila, B., Ateya, A. A., Abd El-Latif, A. A., Almousa, M., & Abd-El-Atty, B. (2025). Brute-force attack mitigation on remote access services via software-defined perimeter. *Scientific Reports, 15*(1), 18599.
24. Shao, W. (2026, March). Interpretable Ensemble Learning for Network Traffic Anomaly Detection: A SHAP-based Explainable AI Framework for Embedded Systems Security. In 2026 14th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-4). IEEE.
25. Shonubi, J. A., & Adelere, M. A. (2025). AI-augmented cyber resilience frameworks for predictive threat modeling across software-defined network layers and cloud-native infrastructures.
26. Su, D., & Zhang, H. (n.d.). Developing a Data-Driven Computational Framework for Scalable Special Education Practices for Autism and Intellectual Disabilities in the US Public School System.
27. Wang, S., Feng, Y., & Fang, X. (2026). A Large Language Model-Enabled Multi-Agent Collaboration Method for Complex Task Solving.
28. Wang, Y., Wen, Y., Wu, X., Wang, L., & Cai, H. (2025). Assessing the Role of Adaptive Digital Platforms in Personalized Nutrition and Chronic Disease Management.
29. Yang, J. (2026). Computational Analysis of How Digital Non-Clinical Communication Tools Influence Social Participation Among Older Adults in Community-Based Elderly Care. Available at SSRN 6682838.
30. Yin, J., Rao, H., & Huang, Y. (2026, March). Dynamic Modeling and Heterogeneity Analysis of Platform User Behavior Time Series. In 2026 International Conference on AI in Education Technology and Applications (AIETA) (pp. 30-33). IEEE.
31. Zhang, Z. (2026). A Study on the Identification of Manipulative Design in Subscription and Payment Interfaces of Digital Consumer Platforms and Its Behavioral Effects. Available at SSRN 6734760.
32. Zhang, Z., Tong, Y., & Gao, Y. (2026). Retrieval-Augmented Generation with Low-Latency Deployment for Vertical Domains Question Answering: A Case Study on Economic Resource Platforms.