International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2, 2026

Peer Reviewed, Referred & Indexed Journal

REAL TIME BANK TRANSACTION FRAUD DETECTION USING KAFKA AND MACHINE LEARNING

[1]MS.THOTA ANITHA, [2]MEDIPALLI ANVES, [3]BANOTH DIVYA, [4]GATHE BHARGAVI, [5]MOHAMMAD ASLAM

[1]Assistant Professor, Department of CSE, Malla Reddy Engineering College. Hyderabad, Telangana

[2,3,4,5]Students, Department of CSE, Malla Reddy Engineering College. Hyderabad, Telangana

ABSTRACT

The rapid expansion of digital banking and online financial services has significantly increased the volume and velocity of transaction data, making fraud detection a critical challenge for modern financial systems. Traditional fraud detection mechanisms are often batch-based, reactive, and unable to process high-frequency transaction streams in real time, resulting in delayed responses and increased financial risks. To address these limitations, this project proposes a Real-Time Bank Transaction Fraud Detection System using Apache Kafka and Machine Learning (ML), which combines real-time data streaming with intelligent predictive analytics. The proposed system utilizes Apache Kafka, a distributed streaming platform, to handle continuous transaction data flow through a producer-consumer architecture. The producer publishes transaction records to Kafka topics, while the consumer continuously retrieves the streaming data and forwards it to the ML model for analysis. The machine learning component, implemented using algorithms such as Random Forest, is trained on historical transaction data to identify patterns of normal and fraudulent behavior. Data preprocessing techniques are applied to convert non-numeric attributes into numerical format and normalize features, ensuring compatibility with ML models. Once deployed, the system performs real-time fraud detection by analyzing incoming transaction streams and classifying them as either legitimate or fraudulent. The integration with a Flask-based web interface allows users to generate Kafka streams, consume data, and visualize prediction results dynamically. Experimental results demonstrate the system's ability to process streaming data efficiently and provide instant predictions, making it scalable and suitable for real-world banking applications. Overall, the proposed approach enhances fraud detection accuracy, reduces response time, and improves financial security by enabling proactive and real-time decision-making.

Keywords: Real-Time Fraud Detection, Apache Kafka, Machine Learning, Random Forest, Data Streaming, Financial Security, Anomaly Detection, Producer-Consumer Architecture, Flask Web Application, Big Data Analytics

I.INTRODUCTION

The rapid growth of digital banking and online payment systems has transformed the financial landscape, enabling fast and convenient transactions across the globe. However, this advancement has also led to a significant increase in financial fraud, posing serious threats to both customers and financial institutions. Fraudulent activities such as unauthorized transactions, identity theft, and money laundering have become more sophisticated, making detection increasingly challenging. Traditional fraud detection systems are typically rule-based and operate in batch mode, which limits their ability to detect anomalies in real time [1], [2]. As transaction volumes grow, there is a critical need for intelligent systems capable of processing large-scale data streams and identifying fraudulent behavior instantly to prevent financial losses [3].

To overcome these limitations, modern approaches leverage real-time data processing and machine learning techniques. Technologies such as Apache Kafka enable high-throughput, fault-tolerant data streaming, allowing financial transactions to be processed as continuous data flows rather than static datasets [4], [5]. In this architecture, Kafka acts as a distributed messaging system with producers and consumers, ensuring efficient data delivery and scalability. On the other hand, machine learning algorithms can analyze transaction patterns, learn from historical data, and detect anomalies that indicate potential fraud [6], [7]. These intelligent models improve detection accuracy and adapt to evolving fraud techniques, making them more effective than traditional systems [8].

In this context, the proposed project, Real-Time Bank Transaction Fraud Detection using Kafka and Machine Learning, aims to integrate streaming technology with predictive analytics to build a robust fraud detection framework. The system processes live transaction data using Kafka and applies machine learning models such as Random Forest for classification [9], [10]. A

web-based interface is used to visualize results and manage data flow, ensuring ease of use and real-time monitoring. This approach provides a scalable, efficient, and proactive solution for fraud detection, enhancing security and trust in digital banking environments while enabling faster and more accurate decision-making [11], [12].

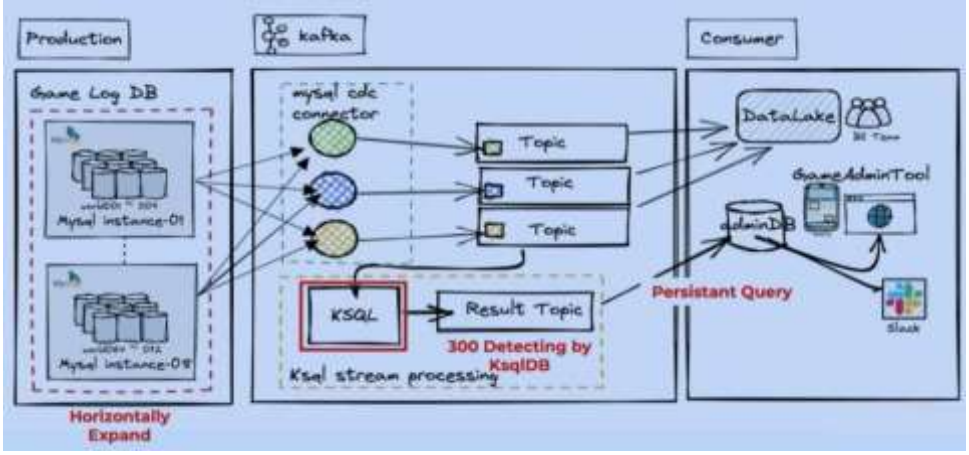


Figure1: System Architecture

This figure illustrates a real-time data streaming architecture using Apache Kafka, showing how data flows from production systems to consumers through stream processing. The architecture begins on the Production side, where multiple MySQL database instances (Game Log DB) store transactional data. These databases are horizontally scaled, meaning multiple instances handle large volumes of data. A MySQL CDC (Change Data Capture) connector continuously monitors database changes (inserts, updates, deletes) and streams these changes into Kafka. In the middle layer, Apache Kafka acts as the core streaming platform. The CDC connector publishes data into multiple Kafka topics, which act as data channels. These topics store real-time event streams that can be consumed by downstream systems. Additionally, KSQL (Kafka Stream Processing Engine) is used to process streaming data in real time. It performs operations such as filtering, aggregation, and fraud detection logic, and then outputs processed data into a Result Topic. The mention of "300 detecting by KsqlDB" indicates real-time analytics or anomaly detection happening within the stream. On the Consumer side, multiple systems subscribe to Kafka topics. Data is sent to a Data Lake for storage and analytics, used by BI teams. It is also sent to an Admin Database (adminDB) and applications like GameAdminTool for monitoring and management. Notifications or alerts can be triggered through tools like Slack.

II SURVEY OF RESEARCH

The study by Daksa et al. (2025) [1] presents a Kafka-based real-time fraud detection system integrated with Apache Spark and Apache Flink for stream processing. The methodology evaluates latency and throughput using machine learning models such as Random Forest for classification. Results indicate that Spark achieved lower latency (~0.8–0.9 seconds) compared to Flink while maintaining stable throughput under varying loads. The study highlights the importance of selecting appropriate stream processing frameworks for real-time fraud detection. However, it mainly focuses on performance benchmarking rather than improving detection accuracy. This research is highly relevant as it demonstrates how Kafka can act as a backbone for scalable fraud detection systems and supports the use of streaming architectures in financial applications.

The work by Dev and Usha (2025) [2] introduces an event-driven fraud detection pipeline using Kafka, KSQL, and Apache Flink. The methodology combines rule-based filtering with machine learning-based anomaly detection to identify suspicious transaction patterns such as abnormal login behavior and velocity violations. The system achieves a high true positive rate of 94.2% with sub-second latency, proving its efficiency in real-time environments. However, the architecture introduces complexity due to multiple integrated technologies. This research emphasizes the effectiveness of combining stream processing and ML models for proactive fraud detection, aligning closely with the proposed system.

Liu et al. (2025) [3] proposed a big data-driven fraud detection framework integrating Apache Kafka, Spark, and machine learning algorithms. The methodology involves feature engineering and training models such as Logistic Regression, Decision Trees, and Random Forest for binary classification. The results demonstrate over 99% accuracy, highlighting the power of

combining big data platforms with ML techniques. However, the study relies on synthetic datasets, which may not fully represent real-world complexities. This research provides strong evidence that scalable architectures with ML integration can significantly improve fraud detection performance.

Veluru et al. (2019) [4] explored a real-time fraud detection system using Kafka and machine learning, focusing on anomaly detection in high-velocity transactional data. The methodology involves continuous ingestion of transaction streams through Kafka and applying ML models to detect unusual patterns. The results show improved fraud detection capability due to real-time processing and continuous learning from historical data. However, the system requires careful tuning of models and infrastructure. This study highlights the importance of real-time analytics and continuous learning, which is a key component of the proposed system.

Dyapa (2025) [5] investigated real-time fraud detection using Kafka and stream processing frameworks, demonstrating that financial organizations can reduce fraudulent transactions by up to 35% using advanced streaming platforms. The methodology focuses on integrating Kafka with machine learning models to process live transaction data and identify anomalies instantly. The results confirm that real-time systems significantly outperform traditional batch-processing systems. However, the study does not deeply explore model optimization techniques. This research reinforces the importance of real-time streaming and immediate response mechanisms in fraud detection systems.

Uzuegbu (2025) [6] proposed an intelligent real-time fraud detection system using machine learning and API-based event streaming with Kafka. The methodology utilizes ensemble learning models such as Random Forest, Gradient Boosting, and Neural Networks to improve detection accuracy. Data preprocessing techniques like normalization and SMOTE are used to handle class imbalance. The system achieved an accuracy of 99.21%, demonstrating high effectiveness. However, the implementation complexity and computational cost are notable limitations. This study strongly supports the integration of ensemble learning and streaming architectures, which enhances the robustness and scalability of the proposed system.

III. WORKING METHODOLOGY

The proposed Real-Time Bank Transaction Fraud Detection System using Kafka and Machine Learning follows a structured, event-driven pipeline to ensure continuous monitoring and instant fraud detection. The process begins with data generation and ingestion, where transaction data is either collected from banking systems or simulated datasets. This data includes both numerical and categorical attributes such as transaction amount, location, account details, and transaction type. Before processing, the dataset undergoes data preprocessing, where non-numeric attributes are converted into numerical format using encoding techniques, and normalization is applied to scale the features. This step ensures compatibility with machine learning algorithms and improves model performance. The next stage involves machine learning model training, where algorithms such as Random Forest are trained on historical transaction data to distinguish between normal and fraudulent transactions. The dataset is split into training and testing sets (typically 70%–30%), and evaluation metrics such as accuracy, precision, and recall are calculated to assess model performance. Once trained, the model is integrated into the real-time pipeline. Simultaneously, Apache Kafka is configured as the core streaming platform, consisting of producers and consumers. The producer publishes transaction data to Kafka topics, while the consumer continuously listens for incoming data streams. In the real-time execution phase, the Kafka consumer retrieves streaming transaction data and passes it to the trained ML model for prediction. The model classifies each transaction as either “Normal” or “Fraud” in real time. The results are then displayed through a Flask-based web interface, where users can generate data streams, monitor transactions, and view prediction outcomes dynamically. Additionally, the system supports continuous data flow, ensuring scalability and low latency. This methodology enables real-time fraud detection, high throughput processing, and proactive decision-making, making it suitable for modern financial systems where immediate response to fraudulent activity is critical.

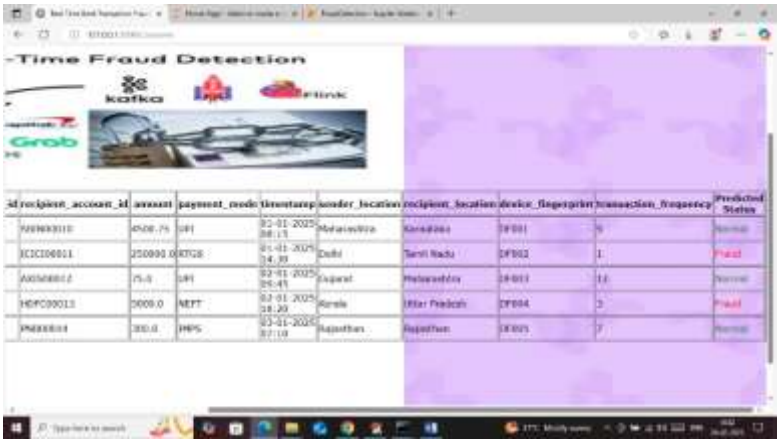
IV RESULTS EXPLANATIONS

Figure 1: Kafka Producer Stream Generation Interface

This figure shows the Kafka Producer Stream Generation Interface of the real-time fraud detection system. In this screen, the user interacts with a Flask-based web application that allows them to initiate the streaming of transaction data into Apache Kafka. The interface contains navigation options such as "Generate Kafka Producer Stream," "Consume Data & ML Fraud Detection," and "Logout," enabling smooth workflow control. When the user clicks on the producer option, the system publishes transaction records to Kafka topics. The message "Kafka producer publish total records = 5" indicates that five transaction entries have been successfully streamed into the Kafka pipeline. These records simulate real-time banking transactions. The producer plays a crucial role in ensuring continuous data flow, which is essential for real-time analytics. This interface simplifies the process of data streaming and demonstrates how Kafka acts as a backbone for handling high-throughput financial data in fraud detection systems.

Figure 2: Kafka Consumer and ML Prediction Output

This figure represents the Kafka Consumer and Machine Learning Prediction Output Interface. In this stage, the Kafka consumer continuously listens to the incoming data streams published by the producer. Once the data is received, it is passed to the trained machine learning model, such as Random Forest, for real-time fraud prediction. The interface displays the transaction records along with an additional column showing the prediction results. By scrolling to the right, users can view whether each transaction is classified as "Normal" or "Fraud." This real-time classification enables immediate identification of suspicious activities. The system demonstrates end-to-end functionality where streaming data is processed instantly and analyzed using ML algorithms. This interface highlights the effectiveness of integrating Kafka with machine learning, providing a seamless pipeline from data ingestion to actionable insights, which is critical for preventing financial fraud in real-world banking systems.



In the above system, the final output clearly demonstrates the integration of machine learning with real-time data streaming using Apache Kafka. After the transaction data is produced and consumed through Kafka, the trained ML model processes each incoming record and classifies it as either "Normal" or "Fraud", which is displayed in the last column of the output table. This classification is based on patterns learned during the training phase. Initially, Jupyter Notebook is used to load the dataset, preprocess it (convert non-numeric to numeric, normalization), and train the model using algorithms such as Random Forest. Once the model is trained, it is integrated into the streaming pipeline. Kafka plays a key role by enabling continuous data flow through its producer-consumer architecture, where the producer sends transaction data and the consumer receives it in real time. The consumer then forwards the data to the ML model for prediction. The Flask-based web interface acts as a user-friendly layer where users can trigger data streaming, view consumed data, and observe prediction results instantly. This end-to-end pipeline ensures real-time fraud detection, scalability, and efficient processing, making the system highly suitable for modern banking environments where quick decision-making is essential.

V.CONCLUSION

The proposed Real-Time Bank Transaction Fraud Detection System using Kafka and Machine Learning provides an efficient and scalable solution to address the growing challenges of financial fraud in digital banking environments. By integrating Apache Kafka for real-time data streaming with machine learning algorithms for intelligent prediction, the system enables continuous monitoring and instant classification of transaction data. Unlike traditional batch-based systems, this approach ensures low latency and immediate detection of suspicious activities, thereby reducing potential financial losses. The implementation demonstrates how transaction data can be streamed through Kafka using a producer-consumer architecture, processed in real time, and analyzed using a trained ML model such as Random Forest. The use of Jupyter Notebook for model training and preprocessing ensures that the system is built on a reliable and well-optimized dataset. Furthermore, the integration of a Flask-based web interface allows users to easily generate streams, consume data, and visualize prediction results, making the system user-friendly and interactive. Overall, the project highlights the effectiveness of combining stream processing with machine learning to build a proactive fraud detection system. It offers high scalability, adaptability to evolving fraud patterns, and real-time decision-making capabilities. Future enhancements may include using larger datasets, advanced deep learning models, and incorporating additional features such as anomaly scoring and alert systems to further improve detection accuracy and system performance.

REFERENCES

[1] N. Leavitt, "Will NoSQL databases live up to their promise?," *Computer*, vol. 43, no. 2, pp. 12–14, Feb. 2010.

[2] T. White, *Hadoop: The Definitive Guide*, 4th ed. Sebastopol, CA, USA: O'Reilly, 2015.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[4] N. Garg, *Apache Kafka*. Birmingham, U.K.: Packt Publishing, 2013.

[5] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proc. NetDB*, 2011, pp. 1–7.

[6] T. Akidau et al., "The dataflow model: A practical approach to balancing correctness, latency, and cost," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1792–1803, 2015.

[7] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, 2007, pp. 443–448.

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. ACM SIGKDD*, 2016, pp. 785–794.

[10] J. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[11] D. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Improving credit card fraud detection with calibrated probabilities," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 677–685.

[12] A. Dal Pozzolo, O. Caelen, Y. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection," *Expert Syst. Appl.*, vol. 41, no. 10, pp. 4915–4928, Aug. 2014.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[14] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, Feb. 2011.

[15] A. Ng, "Machine learning and AI via brain simulations," *Commun. ACM*, vol. 55, no. 8, pp. 32–35, Aug. 2012.

[16] F. Chollet, *Deep Learning with Python*. Shelter Island, NY, USA: Manning, 2017.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[18] M. Zaharia et al., "Apache Spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Nov. 2016.

[19] J. Kreps, "Questioning the lambda architecture," *Confluent Blog*, 2014.

[20] G. Shakya, *Real-Time Data Streaming Using Apache Kafka*. New York, NY, USA: Springer, 2020.

[21] S. Kulkarni and R. K. Mishra, "Real-time fraud detection using machine learning," *Int. J. Comput. Appl.*, vol. 182, no. 44, pp. 1–6, 2019.

[22] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class distributions," in *Proc. ACM SIGKDD*, 1998, pp. 164–168.

[23] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA, USA: O'Reilly, 2019.

[24] W. McKinney, "Data structures for statistical computing in Python," in *Proc. Python Sci. Conf.*, 2010, pp. 51–56.

[25] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.