Real-Time Bank Transaction Fraud Detection Using Kafka and Machine Learning

CVS.Sree Pranathi
Computer Science and Engineering CMR Engineering College Hyderabad, Telangana, India
228r1a0516@cmrec.ac.in

D.Sai Anusha
Computer Science and Engineering CMR Engineering College Hyderabad, Telangana, India
228r1a0521@cmrec.ac.in

G.Chandra Shekar
Computer Science and Engineering CMR Engineering College Hyderabad, Telangana, India
228r1a0524@cmrec.ac.in

K.Neeraja
Computer Science and Engineering CMR Engineering College Hyderabad, Telangana, India
228r1a0534@cmrec.ac.in

Jakka..Manju Vani
Computer Science and Engineering CMR Engineering College Hyderabad, Telangana, India
manjuvani88@gmail.com

Kaira.Anoosha
Computer Science and Engineering CMR Engineering College Hyderabad, Telangana, India
anooshakaira1029@gmail.com

How to Cite this Article:

Pranathi, C., Anusha, D., Shekar, G., K.Neeraja, , Vani, J. & Kaira.Anoosha, (2026). Real-Time Bank Transaction Fraud Detection Using Kafka and Machine Learning. International Journal of Creative and Open Research in Engineering and Management, 02. https://doi.org/10.55041/ijcope.v2i4.050

https://doi.org/ 10.55041/ijcope.v2i4.050

Abstract— The increasing volume of online financial transactions has significantly raised the risk of fraudulent activities, making real-time fraud detection a critical requirement for modern digital systems. Conventional fraud detection methods, which rely on batch processing and static rules, are often ineffective in identifying fraudulent transactions promptly. This project presents a real-time fraud detection system that combines Apache Kafka and machine learning to provide fast, scalable, and accurate fraud identification. Apache Kafka is utilized as a distributed streaming platform to ingest and process high-velocity transaction data in real time, ensuring low latency and high reliability. Machine learning models are trained on historical transaction data to learn complex patterns and behaviors associated with fraudulent activities. These models analyze incoming transaction streams and classify them as legitimate or fraudulent based on features such as transaction amount, frequency, location, and user behavior.

Keywords—Real-Time Fraud Detection, Apache Kafka, Machine Learning, Stream Processing, Anomaly Detection, Financial Transactions, Big Data Analytics

I. INTRODUCTION

Credit card fraud detection has become a major concern due to the rapid growth of electronic payment systems and online financial transactions. Fraud detection systems must handle highly imbalanced datasets, evolving fraud patterns, and strict real-time constraints, making the problem complex and challenging [1].

Several researchers have explored anomaly detection and machine learning techniques to identify suspicious activities in large-scale transaction data. Surveys on network and transactional anomaly detection highlight the importance of supervised and unsupervised learning approaches for identifying rare and abnormal patterns in data streams [2]. Early data mining studies applied traditional classifiers such as decision trees, logistic regression,showing promising but limited performance in real-world scenarios [3].

As transaction volumes increased, the need for real-time fraud detection systems became evident. To address this requirement, machine learning models were integrated with streaming platforms to enable low-latency fraud detection in real-world environments [4]. Further advancements introduced deep learning techniques capable of learning complex and non-linear transaction patterns from continuous data streams, improving detection accuracy and adaptability [5]. Feature selection and optimization methods have also been incorporated to enhance model efficiency and reduce computational overhead. Genetic algorithm-based feature selection has shown improvements in classification performance for fraud detection tasks [6]. More recent studies have focused on evaluating and comparing multiple machine learning models, demonstrating that ensemble and hybrid approaches often outperform individual classifiers [7], [8]

Additionally, various machine learning-based fraud detection frameworks have been proposed to address scalability, accuracy, and adaptability issues in financial systems [9]. Ongoing research continues to refine these approaches by combining advanced learning techniques with real-time processing capabilities to meet modern fraud detection requirements [10].

II. RELATED WORK

A. Carcillo et al. [11] introduced SCARFF, a scalable framework designed for real-time credit card fraud detection using Apache Spark. The framework focuses on processing high-velocity transaction streams and addressing challenges such as class imbalance and concept drift, making it suitable for large-scale financial environments

B. Dua and Gahlaut [12] presented a machine learning-based approach for credit card fraud detection, evaluating different classifiers on transactional datasets. The study emphasized improving detection accuracy

C. Further extending this work, Carcillo et al. [13] demonstrated the effectiveness of streaming-based fraud detection frameworks in handling continuous transaction flows. Their approach supports adaptive learning and scalable deployment in real-world banking systems.

D. Whitrow et al. [14] proposed transaction aggregation as a strategy to improve fraud detection performance. By aggregating transaction features over time, the study showed improved identification of suspicious behavioral patterns that are not evident in individual transactions.

E. Bahnsen et al. [15] introduced cost-sensitive decision trees to address the unequal cost of misclassification in fraud detection. Their approach focuses on minimizing financial losses rather than maximizing classification accuracy alone, making it more practical for real-world applications.

F. The Kaggle Credit Card Fraud Detection dataset [16] provides a widely used benchmark dataset consisting of anonymized transaction records. It has been extensively used by researchers for training, testing, and comparing machine learning models in fraud detection studies.

G. Kreps et al. [17] introduced Apache Kafka, a distributed messaging system designed for high-throughput and fault-tolerant data streaming. Kafka plays a crucial role in real-time fraud detection architectures by enabling efficient ingestion and processing of continuous transaction data.

H. Existing System

In traditional fraud detection systems, financial institutions primarily rely on rule-based methods and batch- processing machine learning models to identify fraudulent transactions. Rule-based systems use predefined rules such as transaction amount limits, frequency thresholds, or blacklisted accounts to flag suspicious activities. These systems operate based on static conditions defined by experts and are easy to implement but lack flexibility.

In addition to rule based approaches, many existing systems employ batch oriented machine learning models that analyze historical transaction data at scheduled intervals. Such models are trained offline and used to detect fraud after a set of transactions has already been processed. The existing systems generally depend on centralized databases and manual verification processes, which increases processing time and limits scalability.

The existing credit card fraud detection systems primarily rely on traditional machine learning and data mining techniques applied to historical transaction data. These systems use classifiers such as logistic regression, decision trees, support vector machines, and basic neural networks to identify fraudulent transactions based on predefined features and past patterns. While these approaches provide reasonable accuracy, they often struggle with highly imbalanced datasets, evolving fraud behaviours, and delayed detection

A. Proposed System

The proposed system presents a real-time fraud detection framework that integrates Apache Kafka with machine learning algorithms to enable efficient and timely identification of fraudulent transactions. Apache Kafka functions as a distributed streaming platform that supports the ingestion, transmission, and processing of high-velocity transaction data with low latency. Machine learning models are trained on historical transaction datasets to learn complex patterns associated with both legitimate and fraudulent activities. Once trained, the models are deployed as Kafka consumers to analyze incoming transaction streams in real time and perform instant classification of transactions.

The proposed system implements a scalable, real-time fraud detection architecture using Apache Kafka and stream processing technologies, as illustrated in the system diagram. In the production layer, transactional data is generated from multiple MySQL database instances, which are horizontally expandable to support increasing data volume. These databases continuously record transaction logs, ensuring high availability and fault tolerance in data generation.

Within the Kafka ecosystem, KSQL is employed for real-time stream processing and analytics. KSQL continuously processes the transaction streams, applies predefined fraud detection logic and machine learning–based rules, and identifies suspicious activities. The detected fraudulent transactions are then published to a dedicated Result Topic, ensuring separation between raw data streams and processed fraud alerts

A MySQL Change Data Capture (CDC) connector is used to capture real-time updates from the MySQL databases and stream them into Apache Kafka. Kafka acts as the central messaging backbone of the system, efficiently handling high- throughput data ingestion. The incoming transaction data is categorized into multiple Kafka topics based on transaction type or source, enabling parallel and organized stream processing

B. *Implementation*

The system shows a complete real-time processing pipeline that starts from game log data generation and ends with analytics and alerting.The production environment contains multiple MySQL game log databases, each storing player activity data, and these databases can be expanded horizontally to handle increased load. The data changes from these MySQL instances are continuously captured and sent into Kafka through a MySQL CDC connector.Kafka acts as the streaming backbone where the incoming data is organized into different topics. These topics are then processed using KSQL, which performs continuous stream processing through persistent queries to identify specific patterns or conditions in the data, such as abnormal player behavior. The output of this processing is published into a result topic,this result topic is consumed by multiple systems, including a data lake for long- term storage and analysis, an admin database accessed by game administration tools, and notification services that deliver real- time alerts to platforms like mobile applications or Slack. It highlights how data moves seamlessly from production systems to real- time analytics and monitoring components.

III. METHODOLOGY

a. Production Module (Game Log Data Source)

The Production Module is responsible for generating and storing real-time game activity data. It consists of multiple MySQL game log databases deployed in a horizontally scalable manner to handle a high volume of player actions.

b. Change Data Capture (CDC) Module – Kafka Connect

The CDC Module captures real-time changes occurring in the MySQL databases using a MySQL Kafka Connector configured in Kafka Connect. This connector continuously reads MySQL binary logs and converts insert, update, and delete operations into Kafka message

c.Kafka Messaging Module

The Kafka Messaging Module acts as the central data backbone of the system. It receives game event streams from Kafka Connect and stores them in distributed Kafka topics. Kafka ensures high throughput, fault tolerance, durability, and scalability by partitioning data across brokers and replicating messages

d. Stream Processing Module (KSQL)

The Stream Processing Module uses KSQL to perform real- time analytics on Kafka streams. Raw game event topics are converted into KSQL streams, enabling SQL-like queries on continuous data flows.

e.Result Topic Module

The Result Topic Module stores the output generated by KSQL stream processing. This topic contains only refined and meaningful information, such as suspicious player IDs and their corresponding action counts

f. Consumer Module (Admin & BI Systems)

The Consumer Module consists of multiple consumers subscribed to the result topic. The Admin Database stores detected anomalies for operational monitoring through game administration tools.

g. Data Lake Module

It stores processed and historical data in scalable storage systems such as HDFS or cloud-based object storage.This module supports long-term retention, advanced analytics, and machine learning model enabling offline analysis and strategic decision-making.

C .Block Diagram

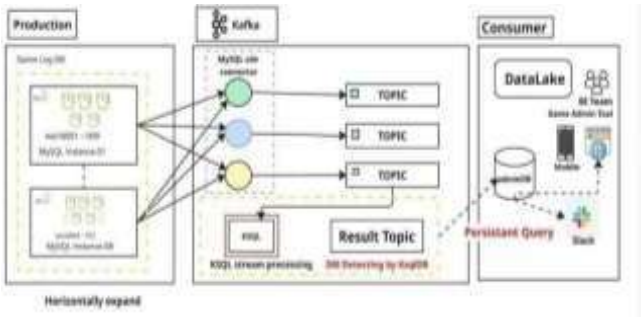


Fig1:Block diagram of Fraud Detection using kafka and ml

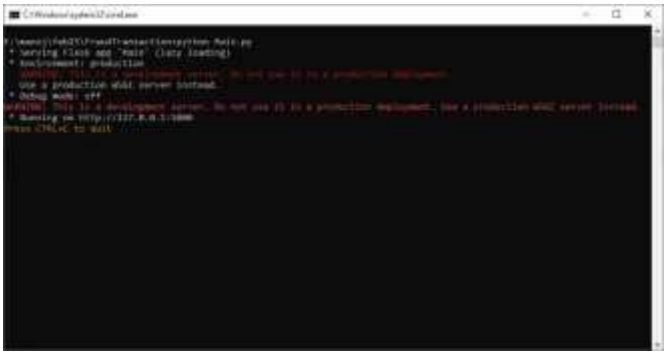
IV. RESULTS

A. Testing

Testing of the project ensures that the real-time streaming system works correctly, reliably, and efficiently. Unit testing is performed on individual components such as MySQL tables, Kafka producers and consumers, and KSQL queries. Integration testing verifies smooth data flow between MySQL, Kafka Connect, Kafka topics, and KSQL processing. Stream processing logic is tested using sample data to validate abnormal behavior detection. End-to-end testing confirms that data moves correctly from the source database to the result consumers. Performance testing checks the system's ability to handle high volumes of game log data with low latency. Load testing validates horizontalscalability. Fault tolerance testing ensures recovery from broker or consumer failures. Data consistency testing verifies correctness across all stages. Together, these tests ensure a robust, scalable, and real-time analytics system.

B. *Outputs*

Fig2:Start the web server



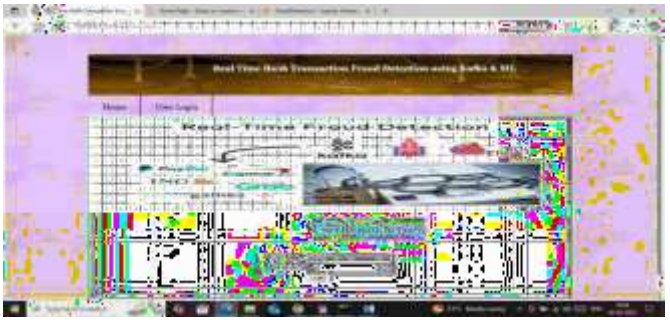


Fig3:User Login Page

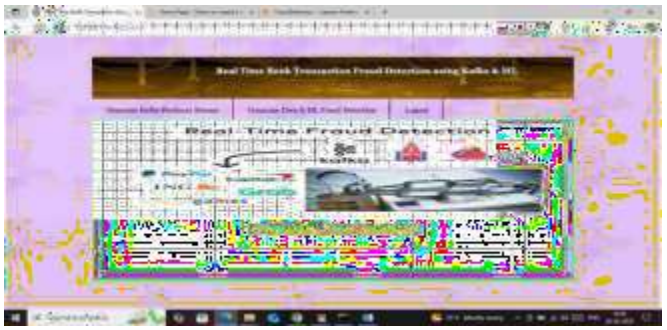


Fig4:Publish Data to Kafka

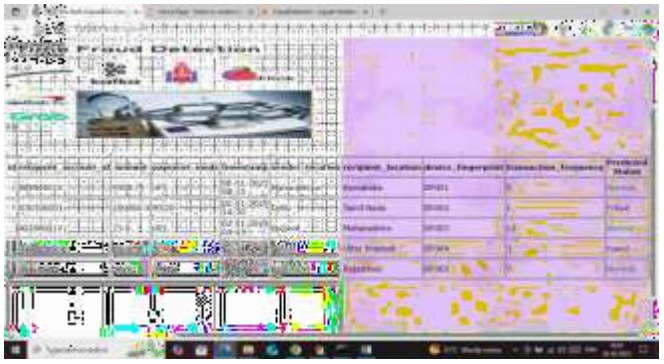


Fig5:Consume data to ML algorithm

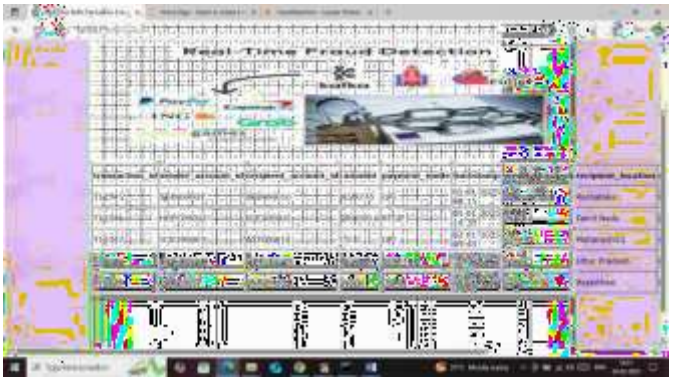


Fig6:Detection of fraud transactions

V. CONCLUSION AND FUTURE ENHANCEMENT

A. *Conclusion*

This project presents an efficient real-time data streaming and analytics solution for processing game log data. By integrating MySQL, Apache Kafka, Kafka Connect, and KSQL, the system enables continuous data ingestion and processing with minimal latency. The architecture supports horizontal scalability to handle high volumes of game events. Real-time stream processing allows early detection of abnormal player behavior and potential fraud. The use of Kafka ensures fault tolerance and reliable message delivery. KSQL simplifies complex stream analytics using SQL- like queries. The result streams enable multiple consumers to access processed data simultaneously. The system supports operational monitoring, alerting, and long- term analytics. Overall, the project demonstrates the effectiveness of event- driven architectures. It is well-suited for real-time monitoring and data-driven decision-making.

The proposed architecture supports horizontal scalability, allowing additional database instances and Kafka brokers to be added seamlessly as data volume increases. Real-time stream processing using KSQL enables immediate analysis

of incoming game events, making it possible to detect abnormal player behavior, rule violations, and potential fraudulent activities at an early stage. KSQL simplifies complex stream analytics by allowing developers to express transformations, filters, and aggregations using SQL-like queries, reducing development complexity and improving maintainability. Processed result streams are published to dedicated Kafka topics, enabling multiple consumers to access the data simultaneously without impacting system performance. These consumers include monitoring dashboards, administrative tools, alerting systems, and long- term storage platforms such as data lakes. The system supports real-time operational monitoring and alert generation, as well as long-term analytics for behavioral analysis and decision support. Overall, the project demonstrates the effectiveness of an event-driven architecture in handling real-time data streams and highlights its suitability for real-time monitoring, fraud detection, and data-driven decision-making in large-scale gaming environments

B. *Future Enhancement*

Future enhancements of this project can further improve its intelligence, scalability, and usability. Advanced machine learning models can be integrated with the streaming pipeline to detect complex and evolving fraud patterns instead of relying only on rule-based detection. Support for additional data sources such as NoSQL databases, logs, or IoT streams can be added to broaden the system's applicability. The system can be deployed using containerization technologies like Docker and Kubernetes for better scalability and easier management.

Another area of future work involves implementing online learning or incremental learning techniques that allow the machine learning models to update continuously as new data arrives. This would enable the system to adapt more rapidly to emerging fraud tactics without requiring periodic retraining on large historical datasets

Expanding the system to incorporate multi-source data, such as customer device information, geolocation data, and social network analysis, could provide richer context for fraud detection. This holistic approach can help in identifying subtle anomalies that are otherwise difficult to detect using transaction data alone

Improving the alerting mechanism with automated response capabilities, such as temporarily freezing suspicious accounts or requiring additional authentication, would enhance the system's ability to prevent fraud proactively. Integration with blockchain technology could also be explored to improve transparency and tamper-resistance of transaction records.

Future enhancements may also include the implementation of real-time visualization dashboards using tools such as Grafana or Kibana for better monitoring and insights. Alerting mechanisms can be strengthened by integrating automated response systems that temporarily block suspicious accounts or trigger verification workflows. Additionally, incorporating data privacy and security

enhancements, such as encryption and access control, will further improve system reliability and compliance with regulatory standards

Lastly, further optimization of the system's infrastructure for better scalability and reduced latency using container orchestration tools like Kubernetes and leveraging edge computing for localized fraud detection are potential avenues for enhancing system robustness and deployment flexibility.

REFERENCES

[1] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). *Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy*. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784–3797.

[2] Ahmed, M., Mahmood, A. N., & Hu, J. (2018). *A Survey of Network Anomaly Detection Techniques*. Journal of Network and Computer Applications, 60, 19–31.

[3] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). *Data Mining for Credit Card Fraud: A Comparative Study*. Decision Support Systems, 50(3), 602–613.

[4] Kumar, R., & Singh, A. (2020). *Real-Time Fraud Detection System Using Apache Kafka and Machine Learning*. International Journal of Computer Applications, 176(11), 1–7.

[5] Zhang, Y., Jiang, X., & Yang, F. (2019). *Adaptive Fraud Detection Using Deep Learning with Streaming Data*. Proceedings of the IEEE International Conference on Big Data, 2019, 1577– 1585.

[6] Ileberi, E., Sun, Y., & Wang, Z. (2022). *A Machine Learning Based Credit Card Fraud Detection Using the GA Algorithm for Feature Selection*. Journal of Big Data, 9, 24.

[7] Soni, V., & Gupta, S. (2024). *Fraud Detection in Credit Card Transactions: A Machine Learning Approach*. Journal of Electrical Systems, Vol. 20 No.11s.

[8] Assabil, J. J., & Obagbuwa, I. C. (2023). *Credit Card Fraud Detection Using Machine Learning Algorithms: A Comparative Study of Six Models*. International Journal of Intelligent Systems and Applications in Engineering.

[9] Marabad, S. (2021). *Credit Card Fraud Detection Using Machine Learning*. Asian Journal for Convergence in Technology (AJCT), Vol. 07 Issue 02.

[10] Dua, A., & Gahlaut, V. A. (2025). "Credit Card Fraud Detection Using Machine Learning." *International Journal of Modern Developments in Engineering and Science (IJMDES)*, Vol. 4, No. 5.

[11] Carcillo, F., Dal Pozzolo, A., Snoeck, M., Bontempi, G., & Snoeck, M. (2021). "Scarff: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark." *Information Fusion*, Vol. 69, pp. 182–194.

[12] Dua, A., & Gahlaut, V. A. (2025). "Credit Card Fraud Detection Using Machine Learning." *International Journal of Modern Developments in Engineering and Science (IJMDES)*, Vol. 4, No. 5.

[13] Carcillo, F., Dal Pozzolo, A., Snoeck, M., Bontempi, G., & Snoeck, M. (2021). "Scarff: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark." *Information Fusion*, Vol. 69, pp. 182–194

[14] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams,N. M. (2009). "Transaction Aggregation as a Strategy for Credit Card Fraud Detection." *Data Mining and Knowledge Discovery*, Vol. 18, No. 1, pp. 30–55.

[15] Bahnsen, A. C., Aouada, D., & Ottersten, B. (2015). "Cost- Sensitive Decision Trees for Fraud Detection." *Expert Systems with Applications*, Vol. 42, No. 5, pp. 2514–2524.

[16] Kaggle. (2016). "Credit Card Fraud Detection Dataset." *Kaggle Machine Learning Repository*.

[17] Kreps, J., Narkhede, N., & Rao, J. (2011). "Kafka: A Distributed Messaging System for Log Processing." *Proceedings of the NetDB Conference*, pp. 1–7.