REAL-TIME BANK TRANSACTION FRAUD DETECTION USING KAFKA ALGORITHM

CH.Sai Sushmitha,M.C.A Student , Amritha sai institute of science and technology, Kanchikacharla (Mandal), A.P- 521180

K.Subash Chandra,Professor , Amritha sai institute of science and technology, Kanchikacharla (Mandal), A.P- 521180

Abstract

Digital banking adoption has accelerated globally, increasing both transaction volumes and the risk of fraudulent activities. Traditional fraud detection techniques, relying on batch processing and periodic analysis, fail to detect fraudulent transactions in real time, leading to financial losses and customer distrust. This paper proposes a **real-time bank transaction fraud detection system** that leverages **Apache Kafka** for streaming transaction data and integrates **machine learning algorithms** such as Random Forest and Naive Bayes for real-time classification. The system architecture ensures scalability, fault tolerance, and low-latency processing. Experimental evaluation on a large-scale bank transaction dataset demonstrates that the proposed system achieves **high accuracy, precision, and recall**, efficiently detecting fraudulent transactions while minimizing false positives. The framework provides a practical solution for modern banking institutions seeking proactive fraud management.

Keywords: Real-time, Fraud Detection, Kafka, Machine Learning, Banking Transactions, Streaming Data.

1. Introduction

The rapid digitization of banking services, including online payments, mobile banking, and e-wallets, has significantly increased financial transaction volume. However, this growth has also expanded the attack surface for fraudsters, making real-time fraud detection essential.

Challenges in traditional fraud detection:

1. **Delayed detection:** Batch processing techniques identify fraudulent transactions after significant delays.
2. **Data imbalance:** Fraud transactions are rare compared to legitimate ones, affecting model performance.
3. **High-volume data:** Banks process thousands of transactions per second, requiring scalable and fast detection systems.

Proposed Solution:

We propose a **real-time fraud detection system** combining **Apache Kafka**, a distributed streaming

platform, with **machine learning classifiers**. Kafka captures live transaction streams, while the classifiers detect anomalies in real time. This combination enables:

- Immediate identification of suspicious transactions
- Minimal latency in alerts
- Scalable and fault-tolerant system architecture

This paper presents the **design, methodology, and evaluation** of this system.

2. Literature Survey

2.1 Machine Learning for Fraud Detection

- **Random Forest:** Widely used due to its robustness against overfitting and ability to handle non-linear relationships. Suitable for imbalanced datasets common in fraud detection.
- **Naive Bayes:** Provides fast probabilistic classification and performs well with large feature sets, although it assumes feature independence.
- **Support Vector Machines (SVM):** Effective for binary classification, but less suitable for real-time streaming due to high computational cost.

2.2 Real-Time Streaming Solutions

- **Apache Kafka:** A high-throughput, fault-tolerant platform for distributed data streaming. Kafka is widely used in financial applications to handle continuous data ingestion.
- **Spark Streaming / Flink Integration:** Stream processing frameworks complement Kafka by preprocessing and transforming data in real time.

2.3 Gaps in Existing Research

Most existing solutions rely on **offline analysis**, which results in delayed fraud detection. Real-time frameworks integrating streaming platforms with machine learning classifiers remain underexplored.

3. Methodology

The proposed system consists of **four key layers**:

3.1 Data Collection Layer

- Transaction data captured from banking platforms, including attributes:
    - Account number, transaction amount, timestamp, transaction type
    - Location, device information, and historical behavior
- Data is sent to Kafka producers for real-time streaming.

3.2 Data Streaming Layer

- **Kafka brokers** receive and store transactions in topics.
- Kafka consumers retrieve messages and feed them into the preprocessing pipeline.

3.3 Data Preprocessing

- Missing values imputation
- Feature scaling and normalization

- Feature engineering: deriving new attributes such as transaction frequency, average transaction amount, and unusual patterns

3.4 Fraud Detection Layer

- **Random Forest Classifier:**
  - Ensemble of decision trees with bootstrap sampling
  - Majority voting for final fraud decision
- **Naive Bayes Classifier:**
  - Probabilistic model computing the likelihood of fraud
- Predictions are combined for **hybrid decision-making**.

3.5 Alert Generation Layer

- Suspicious transactions trigger alerts to the monitoring dashboard and bank security teams.
- Continuous feedback updates models with confirmed fraud cases.

4. System Architecture and Working Procedure

Step-by-step workflow:

1. **Transaction Initiation:** Customers initiate payments or transfers.
2. **Kafka Producer:** Transaction details are sent to Kafka topics.
3. **Kafka Broker:** Manages and stores incoming messages with fault tolerance.
4. **Kafka Consumer & Stream Processing:**
   - Preprocess transactions
   - Extract features
5. **Machine Learning Prediction:**
   - Random Forest and Naive Bayes classify each transaction
6. **Fraud Alert:** Transactions flagged as suspicious trigger real-time notifications.
7. **Model Update:** Periodically retrained with confirmed fraud cases for continuous learning.

Figure 1: Real-Time Fraud Detection Architecture (diagram showing Kafka producer → Kafka broker → Spark streaming → ML model → Alert system)

5. Algorithms Used

5.1 Random Forest

- Ensemble learning method with multiple decision trees
- Robust, reduces overfitting, suitable for large datasets

Steps:

1. Sample data with replacement (Bootstrap)
2. Build multiple decision trees
3. Aggregate predictions via majority voting

5.2 Naive Bayes

- Probabilistic classifier based on Bayes' theorem
- Assumes feature independence
- Computes P(Fraud | Features)

5.3 Apache Kafka

- Distributed streaming platform
- Components: Producer, Broker, Consumer, Topic
- Handles high-volume real-time transaction streams

6. Dataset and Experimental Setup

- **Dataset:** 1 million anonymized banking transactions (simulated with real-world patterns)
- **Features:** Transaction amount, time, type, location, account age, device info
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1-Score, Latency

Experimental Environment:

- Kafka Cluster: 3 Brokers
- Spark Streaming: 2 Executors
- ML Libraries: Scikit-learn for Random Forest and Naive Bayes
- System Latency measured in milliseconds

7. Results and Analysis

Algorithm	Accuracy	Precision	Recall	F1-Score	Latency (ms)
Random Forest	97.5%	95%	92%	93.5%	120
Naive Bayes	94.2%	91%	88%	89.5%	80

Observations:

- Random Forest performed better in detecting fraud with fewer false positives.
- Naive Bayes is faster but less precise.
- Kafka ensured seamless real-time transaction processing.
- Combining both models allows **fast, accurate, and reliable fraud detection**.

8. Conclusion

The proposed system demonstrates that integrating **real-time streaming (Kafka)** with **machine learning algorithms (Random Forest & Naive Bayes)** is effective for bank fraud detection. It achieves high accuracy, low latency, and supports continuous learning. Future improvements include:

- Incorporating deep learning models (e.g., LSTM) for sequential transaction patterns
- Scaling to multi-bank or cross-platform detection
- Enhancing feature engineering with behavioral biometrics

References

1. P. J. R. S. Silva, "Machine Learning Techniques for Fraud Detection in Financial Transactions," IEEE Access, vol. 8, pp. 100234–100245, 2020.
2. A. K. Jain and B. K. Gupta, "Real-Time Data Streaming Using Apache Kafka for Financial Analytics," International Journal of Advanced Computer Science, vol. 12, no. 3, pp. 45–53, 2021.
3. L. Breiman, "Random Forests," Machine Learning, vol. 45, pp. 5–32, 2001.
4. F. Provost and T. Fawcett, Data Science for Business, O'Reilly Media, 2013.
5. R. S. Pressman, Software Engineering: A Practitioner's Approach, 8th Edition, McGraw-Hill, 2014.