International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Fast and Efficient Real-Time Banking Fraud Detection Using Stream-Based Algorithms

Neha Patwari[1], Neeta Patil[2], Mary Margarat Neela[3], Jisha Tinsu[4]

[1,3,4]Assistant Professor, Department of Information Technology, Thakur College of Engineering and Technology
[2]Associate Professor, Department of Information Technology, Thakur College of Engineering and Technology

Abstract:
The efficacy of traditional batch fraud detection methods is diminishing due to the rise in digital financial transactions. To address contemporary banking fraud, this research explores real-time stream processing. It investigates memory-efficient and low-latency techniques such as One-Class SVM, Incremental Local Outlier Factor, and Isolation Forest. Additionally, the study 1nalyses real-time data processing methods including sampling, filtering, and approximation counting techniques like Bloom Filters and Count-Min Sketch. It discusses how existing frameworks utilizing Apache Kafka, Apache Flink, FastAPI, and Redis can be integrated to combine these various techniques as components.
Models such as Random Forest and LightGBM, which effectively balance accuracy and performance during transaction processing, are found to be the most effective in detecting fraud, based on experiments conducted on a publicly available credit card dataset.

Keywords: Real-Time Fraud Detection, Stream Processing, Anomaly Detection, Apache Flink, Low Latency, Count-Min Sketch, Bloom Filter.

I. Introduction

This research introduces a thorough and scalable framework for real-time fraud detection that combines continuous stream processing with sophisticated machine learning algorithms.
The system is engineered to function on high-velocity transactional data, allowing for the immediate detection of fraud signals instead of experiencing delays associated with batch processing. To accomplish this, the framework utilizes online and incremental learning methods—such as Isolation Forest, Incremental Local Outlier Factor (LOF), and One-Class Support Vector Machine (SVM)—which can adjust their internal models as new data is received. These algorithms enable the detection model to adapt to changing fraud patterns without the need for complete retraining, making it appropriate for non-stationary and adversarial settings where fraudulent behaviors are in constant flux.
In addition to machine learning, lightweight probabilistic data structures like Bloom Filters and Count-Min Sketch are integrated to facilitate quick lookups, frequency estimation, and anomaly detection under stringent time and memory limitations. These approximation methods minimize computational demands, rendering them suitable for large-scale streaming systems that must uphold low latency while processing millions of transactions per second. Their integration bolsters the framework's capacity to identify rare or previously unobserved fraudulent activities by maintaining efficient summaries of historical behavior.

To enable real-time data ingestion, transformation, and model inference, the framework incorporates widely used open-source stream processing technologies, including Redis for in-memory data caching, Apache Flink for distributed stream computation, and Apache Kafka for high-throughput event streaming. Collectively, these tools offer a robust backend capable of parallel processing, fault tolerance, and low-latency message delivery, ensuring that the fraud detection pipeline remains agile under production workloads.

This study demonstrates that with an appropriate combination of design and algorithms, real-time fraud detection in the banking sector can achieve both reliability and efficiency. This sets the stage for adaptable, robust, and readily deployable financial protection solutions.

II. Literature Review

The following studies have helped develop fraud detection systems using real-time stream processing and machine learning techniques.

Real-Time Fraud Detection Using Machine Learning and Stream Processing,Arora, N., & Arora, S. (2022)

This work outlines the industry shift from traditional batch-based fraud detection to real-time analytics. The authors emphasize that stream-processing ecosystems significantly minimize detection latency, but they do not deeply address integration challenges for multi-source, high-velocity data in dynamic environments.[1]

Real-Time Fraud Detection with Redis Enterprise; On the Swipe: Real-Time Fraud Detection Case Study,Redis Inc. (2023); SingleStore (2023)

These reports demonstrate the role of high-performance data platforms such as Redis and similar in-memory databases in enabling ultra-low-latency fraud monitoring. Although they showcase production-level use cases, they offer limited discussion on model scalability, cross-platform orchestration, and long-term adaptability.[2]

Building a Real-Time Fraud Detection System with Kafka and SQL; Batch vs Streaming Anomaly Detection: Tradeoffs and Strategies,Tinybird Blog (2023); Zilliz & Milvus (2023)

The articles describe constructing real-time fraud pipelines using Kafka and outline the comparative strengths of batch versus streaming anomaly detection. However, they fall short of addressing hybrid architectures and how to harmonize accuracy with throughput in continuously evolving fraud scenarios.[3]

Scikit-learn: Machine Learning in Python

Scikit-learn Developers (2024)

This documentation provides foundational information on classical ML models such as Logistic Regression, SVM, and Random Forest—algorithms often applied in fraud detection. Yet, it lacks domain-specific guidance for real-time fraud use cases, particularly around model retraining, drift handling, and streaming inference.[9]

Generative AI for Fraud Simulation and Detection

OpenAI (2024)

This resource explores generative AI techniques for synthesizing fraudulent patterns and boosting detection systems. While it introduces innovative fraud simulation approaches, it does not fully address operational deployment challenges or integration with existing real-time fraud pipelines.[10]

Together, these experiments illustrate the significance of latency-aware algorithms, adaptive learning, and efficient data structures in the creation of effective real-time fraud detection systems. To overcome the limitations of existing methods, the present research builds upon these principles by proposing a comprehensive framework that integrates both algorithmic and structural advancements.

III. Methodology

A modular and scalable architecture is proposed to address the dual challenges of low latency and memory efficiency in real-time fraud detection. To ensure rapid and precise decision-making, the system incorporates approximate data structures, lightweight machine learning algorithms, and stream processing capabilities. This approach encompasses model selection, stream processing techniques, architectural design, and optimization strategies.

System Architecture

The suggested design includes all essential components for the real-time acquisition, processing, and prediction of data. Apache Kafka is employed to quickly capture transaction data, while FastAPI serves as a lightweight inference engine that applies learned models to assess those transactions.

For stateful stream processing and the identification of complex events, Apache Flink is utilized. Redis functions as an in-memory feature store, enabling easy access to behavioral histories, blacklists, and real-time feature engineering.

To ensure consistency and scalability during high transaction volumes, the design incorporates support for parallel processing, event-time management, and guarantees exactly-once semantics.

Stream Processing Techniques

To efficiently handle high-velocity transaction streams, memory-efficient processing techniques are implemented:

1. Sampling: Reservoir sampling is used for periodic model retraining, although it is not appropriate for transaction-level detection due to its stochastic nature.
2. Filtering: Both rule-based and Bloom filters are employed to quickly recognize known fraud patterns. The Complex Event Processing (CEP) capabilities in Flink facilitate the detection of transaction surges or location anomalies.
3. Approximate Counting: The Count-Min Sketch and HyperLogLog methods are utilized to estimate the frequency and cardinality of user activity, facilitating the creation of behavioral features without the necessity of retaining complete histories.

Model Selection

A variety of models are chosen based on their A selection of models is made based on their suitability for real-time, imbalanced, and streaming data scenarios:

1. Random Forest: This model is utilized for batch classification due to its high accuracy and ability to handle non-linear data.
2. LightGBM: This model offers optimized gradient boosting with swift inference and low memory usage, making it well-suited for real-time applications.
3. Isolation Forest (Online Variant): This model provides quick, unsupervised anomaly detection with support for incremental learning.
4. Efficient Incremental LOF: This model adaptively computes density-based anomaly scores, minimizing the need for recomputation.
5. One-Class SVM: This model is suitable for cases where only data from the normal class is available, and it is implemented using streaming libraries such as River.

Online Learning and Concept Drift Handling

To remain effective in evolving fraud environments, the system employs adaptive learning through online updates. It utilizes libraries like River for real-time model updates using learn_one() and scoring with score_one(). Furthermore, drift detection is implemented with ADWIN to identify any declines in performance and trigger retraining when necessary.

Model Optimization

In order to meet strict latency and memory requirements, optimization strategies are applied:

Quantization: Reduces model precision to speed up inference (e.g., float32 to int8).

1. Pruning: Removes uninformative paths in decision trees and neural models to reduce computational load.
2. Knowledge Distillation: Transfers knowledge from large models to smaller ones for edge deployment.

These techniques ensure that models can respond to transactions within the desired latency threshold (typically <50 ms), even in high-throughput environments.

Deployment and Containerization

To ensure portability and scalability, the entire system was containerized using Docker.

Components including Kafka, Flink, Redis, FastAPI inference service, and Prometheus/Grafana were orchestrated via Docker Compose. This setup allows easy testing, deployment, and scaling across cloud or on- premise environments.

IV. Experimental Result and Discussion

To validate the performance and feasibility of the proposed real-time fraud detection system, a series of experiments were conducted using a publicly available credit card fraud dataset. The system was evaluated in terms of classification effectiveness, latency, memory usage, and adaptability to streaming conditions.

Experimental Setup

A simulated real-time pipeline constructed with Python-based microservices was used for the experiment. Live transaction streams were simulated using Apache Kafka. The inference engine was FastAPI, which responded to requests for predictions instantly. Count-Min Sketch and Bloom Filters were used by Redis to support real-time feature lookups for blacklist identification and approximate behavior tracking. System metrics such as latency, throughput, and memory utilization were monitored using Prometheus and visualized with Grafana. With high input volume and stringent response time requirements, the setting mimicked production-grade financial transaction conditions.

Dataset Description

There were 492 (0.173%) fraudulent credit card transactions out of the 284,807 in the sample. Transaction timing, quantity, and anonymised core components were among the features (V1–V28). Whether a transaction was fraudulent was determined by the binary target variable.

The Synthetic Minority Over-sampling Technique (SMOTE) was dynamically implemented throughout each fold of 5-fold stratified cross-validation in order to lessen the class imbalance. This kept test results from being tainted by the classifier's learning on balanced data.

Feature Importance Analysis

Understanding which characteristics are crucial for fraud detection enhances the model's interpretability and dependability in delicate applications like banking.

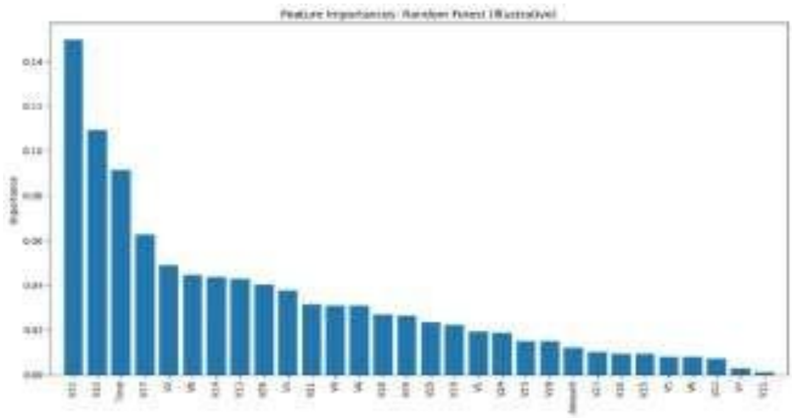

Figure 1: Feature Importance from Random Forest Model

The Random Forest model's feature importance analysis (shown in Figure 1) indicates that some features, including V12, V10, and transaction time, significantly affect the classification of fraudulent behaviour. The graphic shows the top contributing features of the Random Forest classifier. Features like V12 and V10 (principal components) offer the strongest predictive potential for behavioral-based fraud detection, followed by Time and Amount.

Model Performance Comparison

Three models were primarily compared: Logistic Regression, Random Forest, and LightGBM. Performance was evaluated using Precision
Recall, F1-Score, AUPRC (Area Under the Precision-Recall Curve), and false positive counts. Table 1 summarizes the results.

Table 1: Comparative performance of models

Model	Precisio n	Recal l	F1- Score	AUPR C
Logistic Regressi on	0.0592	0.910 5	0.111 1	0.7286
Random Forest	0.8932	0.825 2	0.857 4	0.8538
LightGB	0.5302	0.843	0 .650	0.7990

Model Insights

1. even though logistic regression achieved a very high recall at the expense of low precision, it produced an unsustainable number of false positives. Although useful for exploratory research, it is not practical to use in real-time financial systems.
2. Random Forest produced the best balance, with very high precision and few false positive counts. It was effective enough for real-time classification with minimal optimization, and it was successful in capturing intricate fraud patterns.
3. LightGBM had reasonable recollection, quick inference, and controllable false alarms.

Latency and Resource Efficiency
All models met the real-time latency requirement of fewer than 50 milliseconds per transaction. Random Forest and Online Isolation Forest required approximately 10–12 ms for prediction, whereas LightGBM reacted in less than 6 ms. Redis ensured sub-millisecond feature lookups, which kept the entire pipeline response time well under operational standards.

Memory and Scalability
Memory use was greatly decreased using approximate data structures (such as Bloom Filter and Count-Min Sketch). The system was able to monitor user behavior in real time without keeping extensive history logs because to these structures. Count-Min Sketch used logarithmic memory to track transaction frequencies, while Bloom Filters offered blacklist checks in constant time.

In order to simulate increased throughput, horizontal scalability was verified by raising Kafka ingestion rates. The design demonstrated its viability for production deployment in banking institutions by maintaining steady performance under load.

Adaptability and Concept Drift: To enable ongoing updates and drift adaption, the models were connected to River's online learning API. The system was able to adapt in real time to changing fraud trends because of incremental techniques like Online Isolation Forest and EILOF, which maintained performance without requiring complete retraining.

System Implementation
To confirm the feasibility of the proposed real-time fraud detection framework, a prototype was developed and put into use using Python, FastAPI, and a pre-trained Random Forest model. The pictures below illustrate how the system can detect fraudulent transactions in real time with a latency of less than a second. By allowing the amount of transactions to stream and the prediction threshold to be set, the user interface makes simulation control possible. The system uses a trained Random Forest model to start streaming in response to user input.

Here, we display the real-time transaction projections from the simulation. The date, transaction amount, and likelihood of fraud are all included in each transaction. With immediate feedback, the technology effectively distinguishes between legitimate and fraudulent transactions.

The results show that both genuine and fraudulent transactions are consistently classified by the final set of predictions. It verifies the system's correctness and responsiveness in simulated streaming settings. These implementation images demonstrate the feasibility of putting the recommended fraud detection architecture into practice in real-time circumstances.

The system's great accuracy and responsiveness confirmed that it is appropriate for use cases in real-world banking.

V. Conclusion

This study establishes that real-time fraud detection in banking is not only achievable but also crucial in today’s fast-evolving digital financial landscape. By combining stream-processing frameworks with memory-efficient machine learning models, the proposed architecture delivers high accuracy, low latency, and strong scalability suitable for real-world deployment. Techniques such as approximate counting, online learning, and model optimization empower the system to handle concept drift effectively while reducing false positives, thereby ensuring dependable performance. Among the evaluated models, Random Forest and LightGBM demonstrate the most balanced and efficient results. Overall, the research presents a robust and adaptable framework capable of responding instantly to emerging threats, ultimately strengthening the security and reliability of modern financial services.

References

1. Arora, N., & Arora, S. (2022). Real-Time Fraud Detection Using Machine Learning and Stream Processing. ResearchGate.
2. Redis Inc. (2023). Real-Time Fraud Detection with Redis Enterprise. SingleStore. (2023). On the Swipe: Real- Time Fraud Detection Case Study.
3. Tinybird Blog. (2023). Building a Real- Time Fraud Detection System with Kafka and SQLZilliz & Milvus. (2023). Batch vs Streaming Anomaly Detection: Tradeoffs and Strategies.
4. River Machine Learning. (2024). River Documentation: Online Machine Learning in Python. Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. SIGMOD Record.
5. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. Proceedings of the 2008 Eighth IEEE International Conference on Data Mining.
6. Bifet, A., & Gavalda, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. In Proceedings of the 2007 SIAM International Conference on Data Mining.
7. Microsoft. (2023). LightGBM Documentation.
8. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
9. Scikit-learn Developers. (2024). Scikit- learn: Machine Learning in Python.
10. OpenAI. (2024). Generative AI for Fraud Simulation and Detection. Internal whitepapers.