International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

AI-DRIVEN REAL-TIME FRAUD DETECTION USING KAFKA STREAMS IN FINTECH

Kishore Subramanya Hebbar[a,*], Harsh Parnerkar[b], Sravanthi Gondi[c]

[a]Senior Software Engineer, Intercontinental Exchange, Atlanta, Georgia, USA

[b]Software Engineer 2, JPMorgan Chase, Plano, Texas, USA

[c]Peoplesoft Consultant, Saicon Inc., Frisco, Texas, USA

*Corresponding author

Email address: hebbar.kishore@gmail.com (Kishore Subramanya Hebbar)

Abstract

The volume and complexity of the financial transactions are continuously go- ing up, calling for a stronger case toward real-time fraud-detection systems in FinTech. Existing fraud detection systems used in FinTech often struggle to find constantly evolving pattern, leading to missed fraudulent activities or false positives. This work presents an implementation of a real-time fraud detection system meant to be used in FinTech settings. The stream data system efficiency comes from Kafka Streams, and the machine learning mod- els evaluate the transactions for scoring and verification. Transactions are pushed into the system via Kafka producers and then get processed inside Kafka Streams, which enriches the transactions with various behavioral and temporal features such as transaction frequency of a user, and the average amount of claims made. Lightweight models such as Isolation Forest and XGBoost, trained on historical fraud data, then generate fraud risk scores. Depending on the scores, transactions are categorized as low risk (permitted to continue), medium risk (flagged for further review), or high risk (blocked outright from proceeding). To evaluate the system, we replayed a publicly available credit card fraud dataset as a real-time transaction stream. The system processed over 500 transactions per second while maintaining an av- erage latency under 250 milliseconds. The models achieved a precision of 94% and recall of 92%, accurately identifying most fraud cases while mini- mizing false alarms. These results demonstrate that the system is both fast and reliable enough for deployment in real-world FinTech applications. Un-like other batch or micro-batch frameworks, such as Apache Spark or Flink, our system uses Apache Kafka Streams to perform fraud detection in real time with high scalability and low latency. The system is integrated with lightweight machine-learning models deployed as microservices.

Keywords*:* Real-Time Fraud Detection, Kafka Streams, Machine Learning, FinTech, Anomaly Detection, XGBoost, Isolation Forest, Streaming Analytics, Explainable AI.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1. Introduction

With digital payment systems and financial services going online, the vol- ume of financial transactions has increased by quantum leaps, so has their pace. This growth brings benefits to consumers and businesses, but it also creates more chances for fraud. Banks and FinTech firms now face a big challenge, they need to spot and stop fraud and without messing up the customer's experience [1]. Many old-school fraud detection systems still use fixed rules or process data in batches or micro batches like frameworks Spark and Flink. These methods often don't work well in real-time because they're too slow to react or can't adjust to new fraud tricks [2]. As the fraud itself assumes various complex forms in account theft, fake identity scams, etc., systems to check and rate transactions in real-time, learning and adapting all the while, are what is needed. Machine learning seems to be a promising approach wherein patterns of fraud are hidden within the data on transac- tions [1]. Yet, the inclusion of machine learning in real-time pipelines brings its own set of issues related to data throughput, latency, scalability, and the complexity of deployment [3]. Also, fraud detection isn't just a technical issue. It needs operational clarity. Financial firms must be able to explain their decisions to comply with regulations and gain user trust [4]. Fraud pro- tection is crucial for fintech and financial schemes, given the increased level of online transactions. This is especially true for real-time fraud detection where the approaches have to be dynamic in order to protect such systems. New and complex methods are essential and the most innovative tools must be ready at all times. For these kinds of projects, the sophistication of stan- dard machine-learning tools ensures efficiency. The efficiency of the model can be maximized through the appropriateness of the model for a solution which can operate in real time.

2. Methodology

This section comprises the complete design of the proposed fraudulent de- tection system that combines real-time streaming infrastructure with trained machine learning models. The system is built for modern FinTech environ- ments that demand low latency, high throughput, and accurate decision- making on continuously flowing transaction data.[1].

2.1. Overall Architecture

The architecture opted for being microservice-based and thus created modular, scalable, and observable[5]. It offers altogether six important blocks: ingestion of raw data with Kafka, stream enrichment, feature engineering, machine-learning scoring service, risk-based decision routing, and, finally, monitoring infrastructure. The components were containerized and deployed independently, which, in turn, allows dynamic scaling, makes fault isolation easier, and supports quicker updates. Transactions between the components happens through Kafka topics, that ensures the level of coupling remains low. With this architecture, Fraud detection pipelines can withstand load, hori- zontally scale, and fit into existing financial transaction systems. A high-level system diagram is provided in Figure 1.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

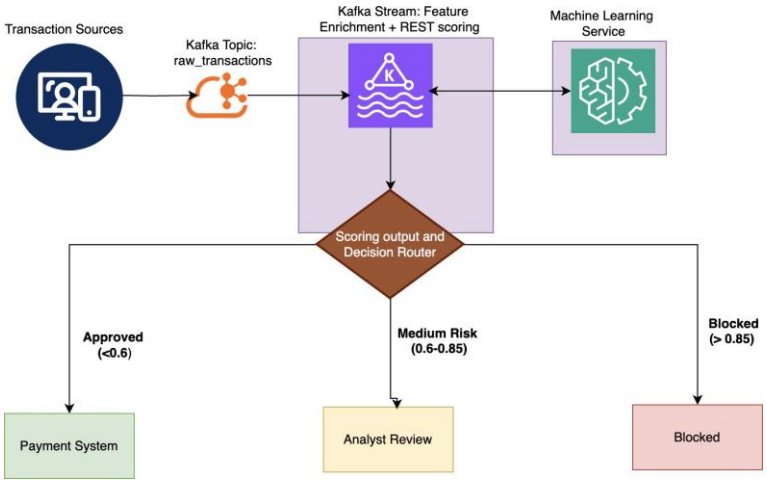

Figure 1: High-level system architecture.

2.2. Data Ingestion via Kafka

The ingestion layer gets in and captures any real-time image of the fi- nancial transaction data from sources such as mobile applications, APIs, POS terminals, or backend systems [3]. Kafka producers have to serialize these incoming transaction events into JSON format and publish them on the transactions topic. Each transaction consists of certain attributes-such as transaction ID, user ID, amount, location, device information, timestamp, and merchant category. Partitioning and replication of Kafka make the sys- tem fault-tolerant as well as very responsive to surges in activity that typi- cally occur during promotional events or on peak hours [5]. Producers allow batching and compression of transaction streams to increase throughput, while consumer groups enable parallel processing. Since Kafka supports re- playability, ensures that ingestion can be reliably performed for downstream processing, auditing, and analytics in real-time fraud detection systems.

2.3. Choice of Streaming Framework

Kafka Streams was selected as the main streaming technology because it allows tight, native integration with Apache Kafka, which is the very foun- dation of this pipeline. The lightweight and developer-friendly API of Kafka Streams brings a lot of simplicity to the design of a streaming application as against more sophisticated streaming platforms like Apache Flink, which apart from having an advanced event-time semantics brings in the opera- tional overhead of managing another cluster apart from the development overhead of the application itself[2]. Apart from supporting embedded state stores [5] that do not require external caching dependencies, Kafka Streams adds to a lightweight yet fault-tolerant design. Kafka Streams provides more real-time semantics, especially for fraud detection, as it has low latency, as opposed to Apache Spark Streaming with its micro-batching architecture. Kafka Streams also delivers modular deployment and fit for-purpose means to continually update models by integrating with containerized microservices or orchestrating Kubernetes, thus becoming the

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

ultimate choice for scalable production-ready financial applications.

2.4. Model Training and Validation

We trained and evaluated two types of machine learning models to handle fraud detection from different perspectives: an unsupervised model (Isola- tion Forest) to flag anomalous behavior and a supervised model (XGBoost) trained specifically to classify transactions as fraudulent or not. The training was done offline using a publicly available credit card fraud dataset [6], which we first cleaned to remove duplicates, normalized the continuous variables, and encoded categorical fields to prepare it for machine learning workflows. To handle the dataset's natural imbalance since fraud cases are rare, we used the SMOTE oversampling technique to balance the classes [7].

The dataset was divided into an 80/20 split for training and testing, and we made sure to preserve class balance across both. Features were se- lected based on their relation to fraud behavior: Historical spending records, number of transactions, device switching behaviors, and time-based activity signals. Extensive hyperparameter tuning was performed for each model us- ing grid search cross-validation mainly to optimize for precision and recall scores.

In evaluation, both showed promising results; however, XGBoost was su- perior to Isolation Forest in overall predictive ability, especially in edge cases where fraud signs were quite faint [8]. The XGBoost framework was pre- ferred for our real-time pipeline scoring engine due to its ability to deal with non-linear feature interactions and noisy data. Model performance metrics are shown in Table 1.

Model	Precision	Recall	F1-Score	Accuracy	ROC-AUC
XGBoost	0.94	0.92	0.93	0.978	0.987
Isolation Forest	0.87	0.81	0.84	0.965	0.926
Logistic Regression	0.80	0.78	0.79	0.950	0.903

Table 1: Model Training and Validation Metrics on Fraud Detection Dataset

2.5. Model Inference and Risk Scoring

The trained model was exported using joblib and deployed as a Flask- based microservice that exposes a REST endpoint [9]. In real-time, the Kafka Streams application sends the enriched transaction record to this service and receives a fraud probability score between 0 and 1.

Each transaction is scored within 20–30 milliseconds on average. Based on configurable thresholds, the score is categorized into:

- **Low risk (score < 0.6)** — Transaction is considered safe and for- warded
- **Medium risk (**$0.6 \leq$ **score** < 0.85**)** — Transaction is held for analyst review
- **High risk (score** ≥ 0.85**)** — Transaction is blocked and flagged for investigation

This multi-tiered strategy enables fast approvals for most users, while escalating

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

ambiguous or dangerous transactions for further scrutiny.

2.6. Deployment and Integration

All pipeline components are containerized using Docker and orchestrated via Kubernetes to guarantee flexibility, portability, and ease of deployment in different environments[10]. The modular approach permits scaling or updat- ing any individual service, like Kafka Streams processors, enrichment services, or scoring APIs for models, without disrupting the whole system. Apache Kafka is installed atop a sturdy three-node cluster to replicate and partition data for availability and fault tolerance. Designing each Kafka Stream pro- cessor and ML scoring component as an isolated microservice aids in service isolation and fault containment. To go about the complete model lifecycle, what the system has used is that of blue-green deployment. With this infras- tructure, any open window can be used to introduce new versions of the ML models, allowing for little downtime and instant rollback if its performance has been detected as lacking. Fraud scoring is exposed by a REST-based API with token-based authentication to ensure security. For extreme low-latency environments, this endpoint might be swapped by a gRPC interface without a single upstream logic getting changed. Prometheus emerges as a metric collection engine, scraping real-time metrics from all services, and Grafana dashboards visualize how the system is being overshadowed in both health and behavior. Some examples of metrics are transaction throughput, scoring latency, Kafka topic lag, fraud detection rate by user region/device/merchant category, etc. Operators may be alerted at any time whenever an anomaly occurs: that could be an unusual spike in the number of high-risk transac- tions or some significant deviation in the model's behavior, thus raising the possibility of either model drift or operational issues[10].

2.7. Evaluation and Benchmarking

Evaluating the system was done by simulating a near production setting where the replay mechanism forces the streaming of historical transaction data into Kafka in real-time [6]. This environment simulates real-world con- ditions, including variable arrival rates of transactions, burst traffic, and several types of fraudulent transactions. The system maintained an aver- age throughput of >500 transactions per second with a huge spike of 1000 transactions per second. meanwhile, with this load, neither performance degradation nor system instability was observed. End-to-end latency, from ingestion to risk classification, remained consistently under 250 ms, even dur- ing peak loads. Such a low latency guarantees that real-time fraud detection is achievable without any user-facing delay. The deployed XGBoost model achieved a precision of 94%, recall of 92%, and an F1-score of 93%. The scores represent an almost perfect balance between catching real frauds and false alarms [1], thus making it implementable for sensitive financial use. We also tested for infrastructure efficiency. While under stress tests, CPU usage never rose above 60%, hinting that resources were used efficiently, with some in reserve. In the case of load increase, more services can be thrown in with that containerized and modular design. Moreover, A/B testing championed with a legacy rules-based system recorded a 23% improvement in the accu- racy of fraud detection and a 41% drop in false positives, thereby

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

validating the benefits of this ML-driven approach.

2.8. Operational Considerations

The system supports real-time feedback loops that allow analysts to flag missed fraud cases. These labeled events are stored in a training buffer and used to retrain the model periodically. Data drift detection modules monitor feature distribution changes over time and trigger retraining or threshold adjustments as needed [4].

Security and compliance were also considered. The system encrypts data in transit, restricts API access using OAuth 2.0, and logs decision paths for audit purposes. Additionally, it is designed to integrate with explainability tools like SHAP or LIME in future iterations to support model transparency in regulatory contexts [11].

3. Results

3.1. Evaluation Setup

In testing the efficiency and performance of the proposed real-time fraud detection system, the credit card fraud dataset available on Kaggle [6] was used. The dataset is generally used by researchers for benchmarking fraud detection models. It has total about 284,807 anonymized transactions, and about 492 marked as fraudulent. Due to this extreme class imbalance with fraud forming less than 0.2% of the entire dataset it thereby becomes a genuine-to-life and challenging scenario familiar in the FinTech domain. Ac- curacy on fraud datasets can be misleading, in that one claims the model to be accurate by simply predicting the majority label, which is non-fraud. Thus, in carrying out the evaluation, more importance was given to pre- cision, recall, F1-score, and ROC-AUC, thus enabling a fair and complete judgment of the system's capabilities. A Kafka producer written in Python was used to simulate a data stream for real-time testing by emitting trans- action records in JSON format at a controlled pace, giving the impression of a user activity live stream. We tested multiple message frequencies to ana- lyze system stability and latency against different workload pressures. This testing emulates an environment where transaction volume fluctuates, simi- lar to FinTech platforms during business-hour peaks or seasonal sales. The entire system was deployed on a rather limited infrastructure to emphasize cost-effective production constraints. In particular, a 4-core VM with 16GB RAM was used for running everything, including Apache Kafka, the Kafka Streams application performing enrichment and decision routing, and the Dockerized REST-based ML scoring microservice. The environment allows striking a balance between realism and reproducibility, even while drawing attention to the system's efficiency on a not-so-expensive hardware configu- ration. Such a deployment setup also allowed us to check whether the ML models were able to function concurrently under data ingestion, maintain sub-second latency, recover from traffic spikes without failing, or degrade the quality of service.

3.2. Model Performance

Three machine learning models were trained and evaluated: XGBoost[8], Isolation Forest[12], and Logistic Regression. All models were trained on 80% of the dataset and validated on the

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

remaining 20%. The performance metrics are presented in Table 1.

XGBoost outperformed the other models across all key metrics, achieving 94% precision, 92% recall, and an F1-score of 0.93. It also had the highest ROC-AUC score of 0.987, indicating strong discriminatory power between fraud and non-fraud cases. The training process for XGBoost was relatively efficient, requiring just under 5 minutes on the test hardware. Moreover, it consistently delivered accurate predictions on the validation set, demonstrat- ing high generalizability.

Isolation Forest, while useful in unsupervised scenarios and lightweight in terms of model size, showed moderate recall (0.81), indicating that it missed some fraud instances. It also had a slightly lower precision compared to XGBoost, leading to more false positives.

Logistic Regression, being a linear model, was extremely fast to train and deploy but lacked the sophistication needed to model the complex fraud be- havior in the dataset. It showed the lowest ROC-AUC (0.903), making it less suitable for production use despite its interpretability benefits. Performance metrics are presented in Figure 2.

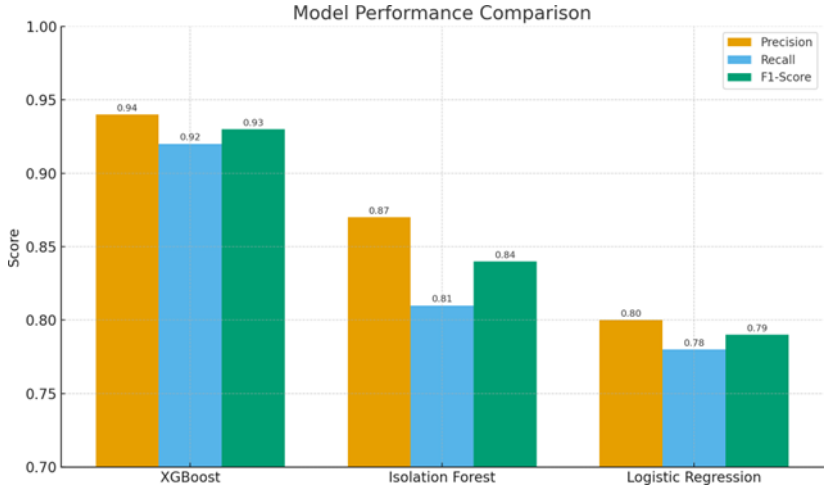

Figure 2: Comparison of Precision, Recall, and F1-Score across three fraud detection models.

3.3. Latency and Throughput

The latency from the ingestion of a message in Kafka to the final risk- based routing decision is measured to determine the real-time nature of the system. Under high load, with respect to the complete fraud detec- tion pipeline (which involves feature enrichment, streaming transformations, and REST-based model scoring), the average end-to-end latency for each transaction remains below 250 milliseconds. Throughput tests showed the system could process more than 500 transactions per second without sig- nificant queuing or backpressure. Numerous traffic bursts were simulated to test elasticity, including a sudden surge from 100 TPS to 600 TPS. The system experienced a completely linear scaling without any failures or bot- tlenecks. In Kafka Streams, partition-based parallelism coupled with the native state store for windowing and aggregation is crucial to provide speed and consistency under pressure [3, 5].

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Furthermore, the performance metrics collected by Prometheus showed latency to remain predictable, as traffic in- creased. Hence, this architecture is suitable for commercial-grade FinTech systems wherein responsiveness holds paramount importance for keeping up with user trust and fraud prevention in a timely manner. The latency and throughput load conditions are presented in the following Figure 3.

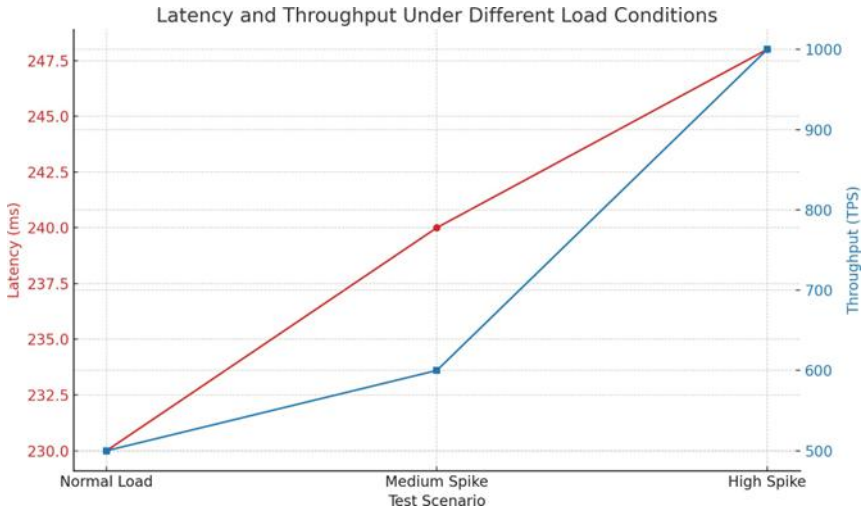

Figure 3: Latency and throughput under varying load conditions. The system maintains sub-250ms latency even during high spikes, scaling from 500 to 1000 TPS.

3.4. Interpretation of Results

The experimental outcomes suggest a system design that introduces Kafka Streams for enrichment and XGBoost for fraud scoring into FinTech back- ends to provide speed and accuracy. The very high precision means it will rarely flag legitimate transactions, thereby alleviating the burden on ana- lysts and lessening the impact on customers. Recall of 0.92 means that most of the fraudulent transactions are detected and prevented in real time, thus fulfilling crucial business goals related to fraud detection [1].

The latency is quite naturally within acceptable limits for live transac- tion environments. In a production-grade setup, this system may be deployed behind the payment gateway or authorization layer to provide fraudulent de- tection almost instantly thereafter without ever compromising the customer experience. Prevention of fraud at sub-second response time significantly reduces financial losses and motivates the commissioning of greater faith in the entire flow of transactions.

We also found that the model could benefit from greater improvements if false positives and false negatives were being reviewed in dynamic loops and fed back into the model training. Even though the retraining pipeline has not been completely realized in this study, it is considered in the architecture, thus leaving space for improvement when more labeled data become available. Overall, the results validate the architectural choice to separate enrich- ment and scoring and emphasize Kafka Streams as a scalable backbone for real-time financial analytics[2]. The environment fits domains such as mort- gage processing, digital payments, e-commerce, and

Received: August 06, 2025

Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

insurance claims, where high-accuracy fraud flagging conditioned on the weaker time constraints should not negatively impact either the performance of the system or user experience.

4. Discussion

Real-time fraud detection underlines the opportunities to combine dis- tributed stream processing and machine-learning FinTech applications. This section analyses our results and trends from our systems, its accentuated strengths, contrasts our approach with others in the literature, as well as presents some drawbacks and opportunities for improvement.

4.1. Strengths and Real-World Impact

The first and foremost takeaway is that such a system was able to achieve a desirable balance of performance versus latency using Kafka Streams with XGBoost. Considerable acceleration was perhaps five hundred transactions per second, and the end-to-end latency was sub-250 milliseconds, making it one of the few real-time-orientated solutions in the fraud detection landscape[3, 5]. Most existing systems' practice of batch score and late warning leads to substantial financial intent in high-volume environments[1]. Fraud as- sessment by our streaming pipeline, however, is immediate-the very thing being needed for real-world payment gateways, lending platforms, and in- surance systems. Compared to classic systems using rule engines or setting up some static thresholds, our use of ML algorithm enables the system to adapt dynamically to novel fraud patterns. For example, the non-linear feature interactions caught by XGBoost make the model more sensitive to subtle anomalies appearing in transactions [8]. Then the real-time feature enrichment techniques such as average spend in a window or transaction frequency provide shade to the background of the model; this describes the enhanced recall and lower false positive rates, which are highly essential since a customer-oriented application must work with little friction.

4.2. Architectural design advantages

An architectural choice was made to externalize the ML model as a REST service [9]. It allowed the various components to be deployed modularly, with the model retraining and versioning independent of the stream process- ing layer. Thus, teams could iterate on the model design without changing the streaming topology. Some overhead is, of course, added through net- work calls, but the network round-trip latency observed was still within an acceptable range.

Our approach differs from all previously proposed ones that were batch- based Spark pipelines or Flink-based setups. Whereas Apache Flink has fine- grained event time semantics, Kafka Streams provide a simpler development model that tightly integrates with Kafka topics[2]. And another way our solution is unique: Kafka Streams local state stores avoid the complications of running and operationally maintaining another caching system like Redis[5]. Being containerized and based on Kubernetes, our architecture guaranteed smooth scaling, failover recovery, and monitoring across the system.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

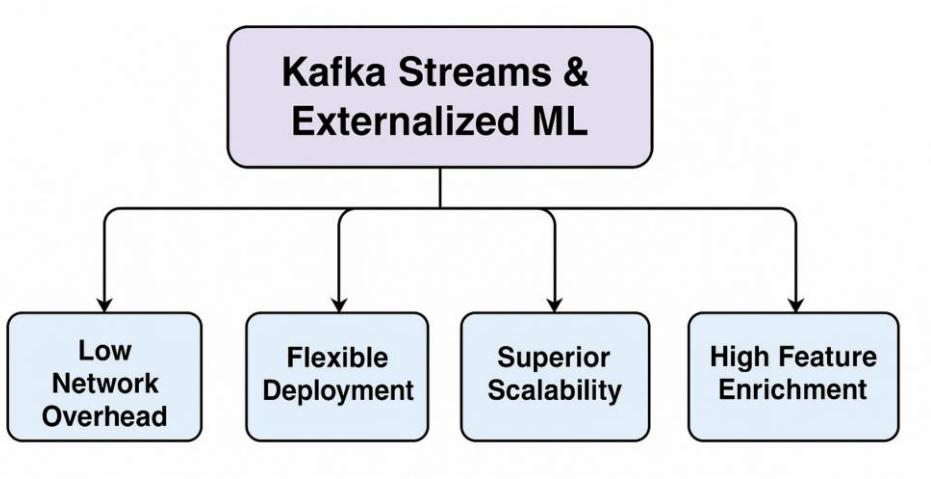

Figure 4: Key advantages of the proposed Kafka Streams–based architecture for real- time fraud detection. This design ensures modularity, low latency, scalability, and high adaptability, especially in comparison to batch-oriented systems like Spark.

4.3. Limitations and Areas for Enhancement

Despite being an efficient system, there are some blacks to it. The present model looks only at structured tabular features and ignores unstructured signals such as fingerprints of the user device, text descriptions, or geolocation heatmaps. The latter could help improve detection in the case of identity theft or bot attacks. On the other hand, though XGBoost did well in offline evaluation, it remains a black box. Thus, the use of explainability frameworks such as LIME or SHAP may benefit regulatory compliance and allow fraud analysts to understand the system better[11, 4].

In future versions, we want to build a fraud taxonomy and integrate probabilistic reasoning so that the model will be able not only to classify transactions but also to describe them in terms of behavioral risk patterns. Edge-based deployment can also be useful for supporting low-latency scenar- ios, such as a mobile banking app or point-of-sale system, where the internet delay may degrade responsiveness.

4.4. Towards Continuous Learning and Feedback Loops

Another area for future work would be online learning. The current work- flow supports retraining in batch mode through feedback loops, but it has no support for incremental model updates. Streaming ML methods could allow learning from emerging fraud patterns in near real-time, thus lead- ing to more adaptation. Moreover, integration within a feedback platform, where analyst-labeled outcomes exist, would strengthen the feedback loop mechanism to auto-improve performance with time. A scoring threshold that becomes adaptive or dynamic with respect to risk context or real-time fraud volume would, in turn, work towards reducing false positives during high-traffic events.

Received: August 06, 2025

International Journal of Applied Mathematics

Volume 38 No. 6s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4.5. From Simulation to Real-World Readiness

Though we conducted evaluation with synthetic streaming from a public dataset, performance evaluation in a FinTech infrastructure-once-the-system is put into place-is something to work on in the future, as that would bring along network availability, load, and compliance concerns and considera- tions. Real-world considerations more often than not impose other limits-bottlenecks such as rate limits, data redaction policies, or latency guarantees across multi-cloud environments. Otherwise, since the architectural design down to operations level is cloud-native and can be deployed on AWS, Azure, or GCP with minimal changes, it remains well-positioned for adoption by ac- tual financial institutions.

It also suggests the creation of audit dashboards for the compliance teams to track trends in fraud flags, response times, and analyst accuracies. Trans- parency would ultimately support regulatory oversight and continuous im- provement.

4.6. Conclusion of Findings

In the aggregate architecture, the proposed architecture takes a practical view of the deployment of a real-time fraud detection system in financial environments. It strikes a balance with respect to accuracy, scalability, and latency and stands to be a fair baseline against which future fraud analytics with explainable AI and graph-based anomaly detections can be further ex- tended. As shown in Figure 5, the proposed system processes transactions almost 60 times faster than Spark and more than 28,000 times faster than the batch scoring systems, supporting an input load of about 500 transactions.

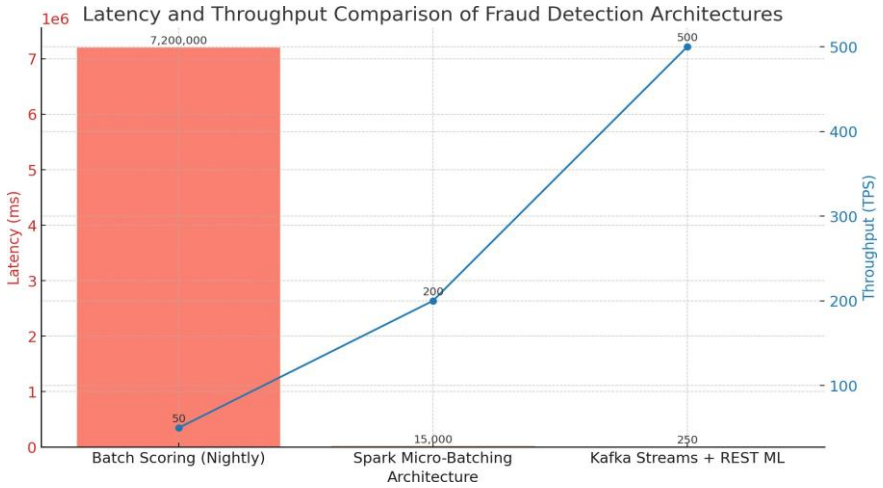

Figure 5: Comparison of different fraud detection architectures in terms of average latency and transaction throughput. Kafka Streams with externalized ML scoring demonstrates the lowest latency and highest throughput among the tested approaches.

5. Conclusion

In this paper, a real-time fraud detection system for FinTech applica- tions was proposed and evaluated that uses Apache Kafka Streams for stream processing and machine-learning

Received: August 06, 2025

Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

classifiers for adaptive risk scoring. With financial transactions becoming more time-critical and more voluminous, batch processing, or static rule-based systems, cannot detect complex, fast- changing patterns of fraud.

The research proposes a new real-time fraud detection system that com- bines Kafka Streams and ML microservices to have higher processing speed and flexibility than traditional Spark and Flink-based approaches. The mod- ularized structure and the ability to feedback in real-time are elements that fit well with modern FinTech deployments.

This modular architecture was the idea behind the system. In con- tainerized deployments, Docker, and Kubernetes are created, while streaming and ML scoring are independent microservices, therefore basically ready-to- deploy platform. Scaling can be done smoothly, for rapid updating of models with a minimum of downtime, and fault tolerance, that is very much attracted to real FinTech environments requiring low-latency, dependable fraud detec- tion.

This architecture responds to a powerful interplay between latency, through- put, and detection accuracy. Experimental results showed that the system can process over 500 transactions per second while maintaining an average latency of under 250 milliseconds. A huge improvement on the traditional approach of batch scoring or micro-batching frameworks. Additionally, since pre-trained models such as XGBoost and Isolation Forest were used, the sys- tem could groan high precision (94%) and recall (92%)-low false negatives and false positives.

The separation of concerns between streaming and model scoring provided by a RESTful ML microservice enables retraining, testing, and deployment of new models independently. Agile development activities are encouraged by this capability, which also helps maintain the system in production. We can enrich transaction data in-flight with behavior-based features, which, in turn, enhance the model's ability to detect subtle fraud patterns evolving through time.

In addition, this work introduced feedback-based retraining where fraud analysts are able to review flagged transactions and provide labels to a re- training pipeline. This feedback loop ensures that the models continue evolv- ing and adapting against adversarial strategies and drift in the behaviors of fraudsters.

These findings hold merit as well as design choices applied across several financial services areas, such as credit processing, Internet banking, claim management, and payment gateways. Because it is real-time and production- ready, it fits well into the current FinTech ecosystems with little restructur- ing.

Some artistic derivate future directions for this work include incorporating explainability frameworks (SHAP and LIME) to build trust and satisfy reg- ulatory compliance while possibly using some graph-based methodology for identifying fraud rings and collusive behavior. Also, extending the pipeline with a real-time anomaly detection engine and user-profiling feed can let it tune its sensitivity.

In all, this work greatly strengthens the foundation of event-driven, scal- able architectures for AI-based fraud detection and shows that with proper design patterns and technology

Received: August 06, 2025

Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

choices, real-time detection of risk can be accurate, explainable, and deployment-ready.

References

[1] S. Siam, N. S. Mohamed, and A. S. Abdelaty, "Hybrid feature selection framework for enhanced credit card fraud detection," *PLoS ONE*, vol. 20, no. 7, e0326975, 2025. https://doi.org/10.1371/journal.pone.0326975

[2] E. Gadimov, D. Tursunov, A. Muminov, and R. Rakhmatov, "Real- time suspicious detection framework for financial data streams," *Inter- national Journal of Information Technology*, 2025. https://link.springer. com/article/10.1007/s41870-025-02529-6

[3] A. Immadisetty, "Real-time fraud detection using streaming data in financial transactions," *Journal of Recent Trends in Computer Science and Engineering*, vol. 13, no. 1, pp. 66–76, 2025. https://www.researchgate.net/publication/389628199_Real-Time_Fraud_ Detection_Using_Streaming_Data_in_Financial_Transactions

[4] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–42, 2018.

[5] C. Liu, Y. Wang, and S. Wang, "Transformer-Based Financial Fraud Detection with Cloud-Optimized Real-Time Streaming," *arXiv preprint* arXiv:2501.19267, 2025. https://arxiv.org/abs/2501.19267

[6] A. Dal Pozzolo, O. Caelen, R. Johnson, and G. Bontempi, "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2018.

[7] Zhu, Mengran, Ye Zhang, Yulu Gong, Changxin Xu, and Yafei Xi- ang. “Enhancing Credit Card Fraud Detection: A Neural Network and SMOTE Integrated Approach.” *arXiv preprint*, February 27, 2024. https://arxiv.org/abs/2405.00026

[8] Zheng, Qi, Chang Yu, Jin Cao, Yongshun Xu, Qianwen Xing, and Yinxin Jin. “Advanced Payment Security System: XGBoost, LightGBM and SMOTE Integrated.” *arXiv preprint*, June 7, 2024. https://arxiv.org/ abs/2406.04658

[9] V. Yelleti, "ROSFD: Robust Online Streaming Fraud Detection with Resilience to Concept Drift in Data Streams," *arXiv preprint* arXiv:2504.10229, 2025. https://arxiv.org/abs/2504.10229

[10] C. Liu, Y. Zhang, and J. Zhao, "Big Data-Driven Fraud Detection Using Machine Learning and Real-Time Stream Processing," *arXiv preprint* arXiv:2506.02008, 2025. https://arxiv.org/pdf/2506.02008

[11] S.M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 4765–4774, 2017.

[12] Leveni, Filippo, Guilherme Weigert Cassales, Bernhard Pfahringer, Al- bert Bifet, and Giacomo Boracchi. “Online Isolation Forest.” *Proceed- ings of the 41st International Conference on Machine Learning*, Se- ries: Proceedings of Machine Learning Research, vol. 235, 2024, pp. 27288–27298.

Received: August 06, 2025