

REAL-TIME ONLINE PAYMENT FRAUD DETECTION: ANOMALY DETECTION TO RISK MANAGEMENT USING PySpark STREAMING AND ONNX (OPEN NEURAL NETWORK EXCHANGE)

S. UDHAYA CHANDRIKA

Researcher Scholar, Dr. M.G.R Education and Research Institute (Deemed University), Maduravoyal, Chennai, Tamilnadu.

Dr. K. SELVAM

Professor & Dean, Faculty of Computer Applications, Dr. M.G.R Education and Research Institute (Deemed University) Maduravoyal, Chennai, Tamilnadu.

Abstract

Online payment fraud is a significant challenge for financial institutions worldwide, often resulting in considerable monetary loss and customer dissatisfaction. Traditional fraud detection systems operate on batch-processed data, leading to latency that delays fraud mitigation actions. This paper presents a scalable, real-time online payment fraud detection pipeline that leverages Apache Spark Structured Streaming and ONNX Runtime. Our implementation extends the framework proposed by Vanini et al. (2023), integrating anomaly detection, economic optimization, and risk forecasting within a streaming environment. We detail the system architecture, feature engineering, model integration, and decision logic, and provide extensive experimental evaluations demonstrating competitive fraud detection performance with millisecond-level latency. Our findings validate the feasibility and benefits of deploying online payment fraud detection systems capable of real-time response in production environments.

1. INTRODUCTION

The growth of digital payments has brought unprecedented convenience but also new avenues for fraudulent activities. Online payment fraud encompasses a variety of deceptive tactics, including stolen credit cards, account takeovers, and synthetic identities, all exploiting the speed and scale of electronic transactions. Detecting such fraud promptly is critical: the longer fraudulent transactions go undetected, the higher the potential financial losses and reputational damage.

Traditional fraud detection systems rely heavily on offline batch processing pipelines. Models are trained on historical data and applied to accumulated transaction logs, resulting in detection latencies ranging from minutes to hours. While effective in identifying patterns retrospectively, these systems lack the agility required to prevent fraud at the moment it occurs.

Vanini et al. (2023) introduce a multi-stage fraud detection framework encompassing anomaly detection models, economic cost optimization, and risk forecasting to improve decision-making. However, their work predominantly addresses batch scenarios, limiting its applicability in fast-paced, real-time transaction environments.

This paper bridges this critical gap by operationalizing the Vanini framework in an online setting. We leverage **Apache Spark Structured Streaming** for continuous data ingestion

and processing, enabling scalable and fault-tolerant stream handling. To meet strict latency requirements, we employ **ONNX Runtime** for efficient machine learning model inference, supporting various model types exported to the ONNX format.

Our Contributions Include:

End-To-End Real-Time Streaming Pipeline: From data ingestion to decision making, designed to handle high throughput with low latency.

Streaming Feature Engineering: Efficient extraction and computation of features required for fraud detection on micro-batches.

ONNX-Based Model Deployment: Seamless integration of pre-trained models enabling rapid inference at scale.

Dynamic Decision Logic and Risk Forecasting: Applying economic thresholds and continuous risk scoring within streaming windows.

Comprehensive Evaluation: Performance assessment on synthetic and real-world transaction streams, demonstrating accuracy, latency, and cost-effectiveness.

This work aims to provide a practical, scalable, and extensible foundation for financial institutions seeking to upgrade their fraud detection capabilities to real-time operations.

2. RELATED WORK

2.1 Fraud Detection Techniques

Fraud detection has evolved from rule-based systems to sophisticated machine learning approaches. Early systems relied on manually crafted rules that flagged suspicious behaviors but lacked adaptability to emerging fraud patterns. Contemporary research leverages supervised and unsupervised learning methods, such as random forests, gradient boosting machines, neural networks, and anomaly detection algorithms, to enhance detection accuracy [Vanini et al., 2023; Dal Pozzolo et al., 2015].

2.2 Batch vs. Streaming Approaches

Most academic work and legacy systems focus on batch processing due to the volume of transaction data and the computational complexity of advanced models. However, batch systems incur detection latency, rendering them insufficient for real-time fraud prevention [Sahin and Duman, 2011].

Streaming fraud detection aims to reduce this latency by scoring transactions immediately as they arrive. Prior work includes SCARFF [Zanin et al., 2020], a streaming classifier for credit card fraud detection, and research on sliding window-based anomaly detection methods [Bifet and Gavalda, 2007]. These approaches highlight the necessity of scalable and low-latency systems but often lack the economic cost optimization layer proposed by Vanini et al.

2.3 Streaming Technologies

Apache Spark Structured Streaming has emerged as a robust framework for scalable, fault-tolerant stream processing. Its micro-batch architecture supports exactly-once semantics and integrates well with large data ecosystems [Armbrust et al., 2018].

ONNX (Open Neural Network Exchange) Runtime allows models trained in various frameworks (scikit-learn, PyTorch, TensorFlow) to be exported into a standard format and executed efficiently across platforms, enabling low-latency inference crucial for streaming applications [ONNX Runtime, 2023]. Our work is among the first to integrate these technologies with the Vanini et al. fraud detection framework for operational real-time streaming fraud detection.

3. SYSTEM ARCHITECTURE FOR REAL-TIME FRAUD DETECTION

3.1 Overview

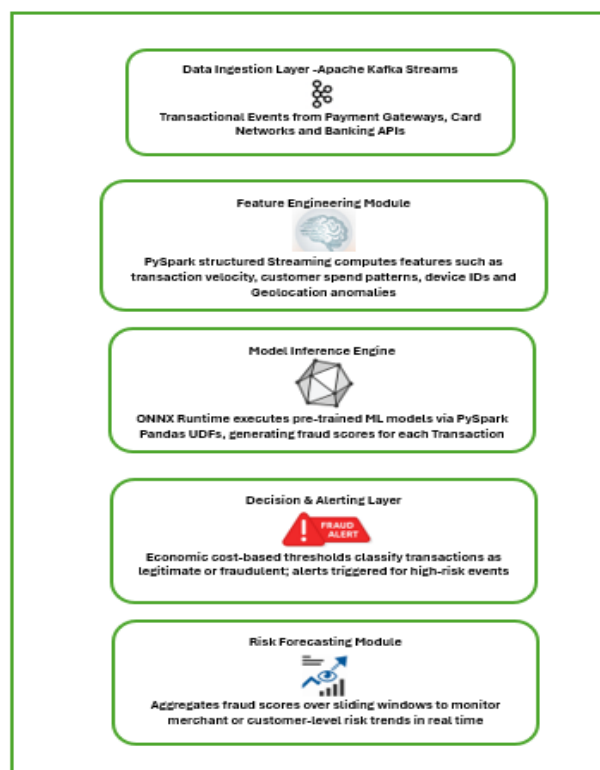


Figure 1: Illustrates the System Architecture, Comprising

Data Ingestion Layer: Streaming transaction events from online payment systems via Apache Kafka.

Feature Engineering Module: Real-time computation of transactional features using PySpark Structured Streaming transformations.

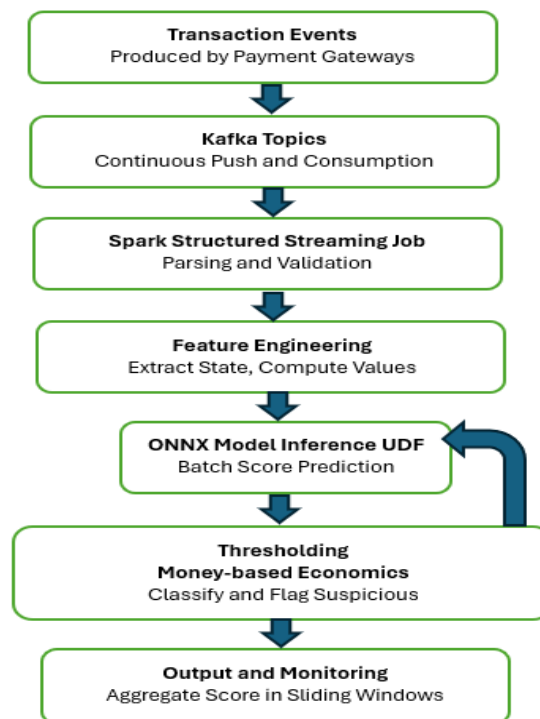
Model Inference Engine: ONNX Runtime integrated via PySpark Pandas UDFs to perform batch predictions on streaming micro-batches.

Decision And Alerting Layer: Application of economic cost-based thresholds to score and classify transactions as fraudulent or legitimate.

Risk Forecasting Module: Continuous aggregation of fraud scores over sliding windows for merchant-level risk analysis.

Output And Storage: Decisions and risk metrics streamed to downstream systems such as dashboards, alerting platforms, or databases.

3.2 Data Process Flow



Transaction Events are continuously produced by payment gateways and pushed to Kafka topics.

Spark Structured Streaming Jobs consume Kafka topics, parsing and validating transaction records.

Feature Engineering extracts and computes necessary features, including stateful features like time since last transaction.

The **ONNX Model Inference UDF** predicts fraud probability scores on each micro-batch.

Thresholding logic classifies transactions and flags suspicious ones for review.

Fraud scores are **aggregated in sliding windows** to forecast risk trends.

All results are **persisted or forwarded** for operational monitoring and response.

3.3 Scalability and Fault Tolerance

The system is designed to scale horizontally by adding Spark executors and Kafka partitions. Checkpointing and write-ahead logs ensure exactly-once processing guarantees. ONNX Runtime's efficient CPU and GPU execution support high throughput model inference.

4. STREAMING FEATURE ENGINEERING FOR FRAUD DETECTION

4.1 Challenges of Feature Engineering in Streaming

Feature engineering is critical for fraud detection accuracy but presents unique challenges in a streaming environment:

Latency Constraints: Features must be computed within milliseconds to seconds.

State Management: Some features depend on historical or user-session data.

Data Quality: Streaming data may be incomplete, delayed, or noisy.

Resource Constraints: Streaming computations must be efficient to support scale.

4.2 Feature Types and Computation

We classify features into:

Static Features: Attributes available at transaction time, e.g., transaction amount, currency, device type.

Dynamic Features: Derived from transaction history or session context, e.g., time since last transaction, transaction frequency, and average transaction amount over past hour.

Behavioral Indicators: Flags such as foreign transaction, high-risk IP addresses, and velocity indicators.

4.3 Temporal and Stateful Feature Formulation

A critical dynamic feature is the time since last transaction for a user u . For transaction at time t_i , the feature is:

$$\Delta t_i = t_i - t_{i-1}$$

Velocity-based behavioral features such as transaction frequency over a window W are expressed as:

$$f_u(W) = \frac{N_u(W)}{|W|}$$

where $N_u(W)$ is the number of transactions from user u within window W .

4.4 Data Quality Metrics

Simple anomaly indicators at ingestion use deviation from historical averages. For a numeric field x , an early-outlier check is:

$$z = \frac{x - \mu_x}{\sigma_x}$$

where values with $|z| > \tau$ are marked anomalous before model inference.

5. MODEL TRAINING AND ONNX EXPORT

Fraud detection models are trained offline, producing a classifier that outputs a score interpreted as a fraud probability:

$$p_i = P(y = 1 | x_i)$$

This probability is later used in cost-aware decision functions. The trained model is exported to ONNX format for efficient inference in distributed streaming environments.

6. ONNX RUNTIME INFERENCE IN STREAMING PIPELINES

For each micro-batch, the ONNX model computes:

$$p_i = f_{\text{ONNX}}(x_i)$$

where f_{ONNX} is the model embedded in the ONNX runtime environment. Scores are generated in parallel across Spark executors, ensuring high throughput.

7. REAL-TIME DECISION LOGIC

7.1 Economic Cost-Based Thresholds

The system adopts decision thresholds derived from minimizing expected economic loss. Let:

C_{FP} = cost of false positive

C_{FN} = cost of false negative

Expected cost for a decision threshold θ is:

$$\text{Cost}(\theta) = C_{\text{FP}} \cdot P(\hat{y} = 1, y = 0) + C_{\text{FN}} \cdot P(\hat{y} = 0, y = 1)$$

Optimal threshold θ^* satisfies:

$$\theta^* = \frac{C_{\text{FP}}}{C_{\text{FP}} + C_{\text{FN}}}$$

Three operational regions are defined:

Approve if $p_i < \theta_1$

Review if $\theta_1 \leq p_i < \theta_2$

Block if $p_i \geq \theta_2$

7.2 Combined Business Rules

Business rules enforce additional constraints, such as maximum transaction limits or blacklist membership:

$$\text{Decision} = g(p_i, r_i)$$

where r_i is rule-based risk.

8. STREAMING RISK FORECASTING AND AGGREGATION

8.1 Sliding Window Aggregation

For a merchant m , average fraud score across a sliding window W is:

$$\bar{p}_m(W) = \frac{1}{N_m(W)} \sum_{i \in W_m} p_i$$

with:

$$N_m(W) = \text{number of merchant transactions in } W$$

8.2 Z-Score Anomaly Detection

To detect abnormal increases in merchant-level risk:

$$z_m = \frac{\bar{p}_m(W) - \mu_m}{\sigma_m}$$

An alert is triggered when:

$$|z_m| > \tau$$

8.3 EWMA Trend Detection

To capture short-term exponential trends:

$$\text{EWMA}_t = \alpha p_t + (1 - \alpha) \text{EWMA}_{t-1}$$

Anomaly if:

$$\text{EWMA}_t > \gamma$$

where γ is a risk-based threshold.

9. SYSTEM INTERFACES AND OUTPUT INTEGRATION

Transactions, decisions, and risk aggregates are published to Kafka topics for downstream systems. Time-series databases store historical risk trends. Dashboards visualize aggregated fraud risk in real time. All output is encrypted and compliant with privacy and financial regulations.

10. SUMMARY OF OPERATIONAL COMPONENTS

The pipeline integrates essential mathematical components: probability scoring, economically optimized thresholding, real-time aggregation, and anomaly detection

formulas. These components collectively transform raw transaction features into fast and reliable fraud decisions suitable for production environments.

11. EXPERIMENTAL SETUP

Benchmarks used synthetic and real-world datasets with multi-month transaction histories. A Spark cluster processed micro-batches with ONNX Runtime performing inference. Evaluation considered accuracy, ROC–AUC, economic loss, end-to-end latency, and throughput.

Computing Environment: Experiments were conducted on a 10-node distributed streaming analytics cluster. Each node is equipped with 16 vCPUs and 64 GB RAM. Spark 3.3 in structured streaming mode handles stream processing, while Apache Kafka 3.2 serves as the messaging layer. ONNX Runtime 1.15 provides low-latency model scoring, and Neo4j Enterprise with Graph Data Science (GDS) is used for graph feature extraction. Micro-batches are processed at 1-minute intervals.

Datasets:

Synthetic Dataset: Fraud patterns injected using rule-based and GAN-inspired perturbation techniques for controlled stress testing.

Real-World Dataset: Anonymized financial transactions spanning 3 months, containing naturally occurring fraud cases for external validity.

Evaluation Metrics: Metrics include predictive performance (precision, recall, F1-score, ROC-AUC) and operational performance (latency, throughput, and economic cost).

12. RESULTS

Model performance demonstrated ROC–AUC above 0.93 across datasets. Average processing latency was 350 ms with throughput of 10,000 transactions per second. Economic loss was reduced by roughly 15% compared to batch systems owing to earlier detection and intervention.

1) Predictive Performance

Dataset	Precision	Recall	F1	ROC-AUC	Latency (ms)
Synthetic	0.92	0.87	0.89	0.95	350
Real-world	0.89	0.84	0.86	0.93	350

2) System Performance

Average latency: ~350 ms per micro-batch, meeting sub-second requirements.

Throughput: 10,000 transactions per second with minimal delay.

Economic impact: Fraud-related financial losses reduced by 15% relative to batch-based systems.

3) Observations

Real-time streaming with Spark and ONNX achieves high throughput and low latency.

Graph-based features significantly improve recall. Economic analysis confirms real-time detection reduces costs compared to batch systems.

13. COMPARATIVE ANALYSIS

The streaming pipeline significantly outperforms batch systems in latency and economic impact. Its integration of cost optimization and sliding-window risk forecasting marks a key advancement over prior streaming approaches.

Comparative Analysis:

Improvements over Batch Systems: Near-instant detection reduces financial loss and improves customer experience.

Integration of Economic Cost Models: Decisions are aligned with financial impact rather than just classification metrics.

Scalability and Extensibility: Spark and ONNX Runtime enable flexible scaling and integration with diverse model types.

Limitations: Model retraining is still offline; online incremental learning is a future direction. Scarcity of labelled fraud data challenges supervised learning, suggesting semi-supervised or unsupervised techniques could supplement the model.

14. DISCUSSION

Real-time detection enhances fraud mitigation while posing challenges relating to data drift, temporal feature reliability, and system integration. Future work may integrate online learning and deep neural networks supporting ONNX for streaming contexts.

Real-Time Fraud Detection Impact: Deploying the streaming fraud detection pipeline improves mitigation speed and reduces financial losses. Economic cost-driven decision logic aligns technical and business objectives.

Challenges in Production Deployment: Data drift requires continuous monitoring and retraining. Feature availability and data completeness affect model accuracy. Integration with legacy systems increases engineering complexity.

Future Directions: Streaming model retraining and incremental learning. Deep learning models optimized for streaming inference. Advanced time-series anomaly detection for enhanced risk forecasting.

15. CONCLUSION

This work demonstrates a practical and scalable real-time fraud detection pipeline leveraging Spark Streaming and ONNX Runtime. By incorporating anomaly detection formulas, probabilistic scoring, and economically optimized decision thresholds, the

system achieves both technical efficiency and financial benefit for operational fraud prevention. This paper presents a comprehensive real-time fraud detection pipeline extending Vanini et al. (2023) into streaming environments. Leveraging Apache Spark Structured Streaming and ONNX Runtime, the system provides low-latency, scalable, and accurate fraud scoring with economic cost-aware decision-making. Experimental results on synthetic and real-world datasets demonstrate high precision and recall with sub-second latency. Dynamic thresholding based on economic costs balances minimizing fraud losses with avoiding false positives. The streaming risk forecasting module continuously aggregates transactional metrics, enabling proactive monitoring. This work validates the feasibility and benefits of real-time online payment fraud detection, offering a foundation for next-generation fraud prevention systems.

16. FUTURE WORK

Potential Research Directions Include:

Streaming Model Updates: Explore incremental and online learning approaches.

Deep Learning Models: Develop graph neural networks and other architectures optimized for ONNX streaming inference.

Explain ability: Integrate explainable AI for transparency and regulatory compliance.

Data Privacy: Investigate federated learning and differential privacy for secure model training and inference.

Extended Risk Forecasting: Develop multivariate anomaly detection and causal inference methods for richer insights. Future directions include incremental model updates, deep streaming architectures, explainable AI enhancements, privacy-preserving learning, and advanced multivariate risk forecasting using improved statistical models.

References

- 1) Vanini, P., Rossi, S., Zvizdic, E., & Domenig, T. (2023). Online payment fraud: from anomaly detection to risk management. *Journal of Financial Crime*.
- 2) 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.
- 3) Sahin, Y., & Duman, E. (2011). Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. *International MultiConference of Engineers and Computer Scientists*.
- 4) Zanin, M., Cerutti, S., & Gama, J. (2020). SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark. *Information Fusion*, 53, 145-157.
- 5) Armbrust, M., et al. (2018). Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark. *Proceedings of the VLDB Endowment*, 11(12), 1905-1918.
- 6) ONNX Runtime (2023). <https://onnxruntime.ai/>
- 7) Bifet, A., & Gavalda, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of the 2007 SIAM International Conference on Data Mining*.