10.48047/jocaaa.2019.26.02.01

ENTERPRISE-GRADE AML THREAT DETECTION USING TIME–FREQUENCY SIGNALS AND SPRING BOOT MICROSERVICES

Naga Charan Nandigama
Independent Researcher, Tampa, Florida, USA

ABSTRACT

The increasing sophistication of money laundering schemes requires financial institutions to adopt advanced analytical systems capable of uncovering hidden, rapidly evolving suspicious activity patterns. Traditional rule-based Anti-Money Laundering (AML) systems often struggle to detect irregular transaction behaviors that manifest across both temporal and frequency domains. This study proposes an enterprise-grade AML threat detection architecture that integrates time–frequency signal analysis, leveraging wavelet and short-time Fourier transform (STFT) techniques for extracting high-resolution behavioral features from financial transactions. These features are processed using distributed analytics pipelines and served through a modular Spring Boot microservices ecosystem, enabling scalable ingestion, anomaly scoring, case creation, and regulatory compliance workflows. Oracle Database is employed as the secure persistence layer to store transactional records, enriched features, audit logs, and AML decision outputs. Experimental evaluations demonstrate improved anomaly detection accuracy, reduced false positives, and enhanced operational flexibility, positioning the proposed system as a next-generation AML compliance solution.

Keywords: AML detection, time–frequency analysis, wavelet transform, Spring Boot microservices, Oracle database, STFT, financial anomaly detection, enterprise compliance systems.

I. INTRODUCTION

Money laundering activities increasingly leverage digital financial platforms, rapid fund transfers, and fragmented transaction chains that make detection difficult for conventional Anti-Money Laundering (AML) systems. Traditional rule-based approaches fail to capture complex behavioral variations and temporal irregularities commonly used to disguise illicit financial flows [1]. As the volume and velocity of financial transactions increase, there is a growing need for architectures capable of analyzing dynamic patterns across multiple dimensions, particularly the time and frequency domains [2].

Time–frequency analysis provides a powerful foundation for modeling non-stationary transaction behaviors, revealing hidden bursts, seasonal shifts, and periodic irregularities that may indicate suspicious activity [3]. Techniques such as the short-time Fourier transform (STFT) and wavelet transforms have demonstrated notable effectiveness in extracting meaningful spectral signatures from financial time series [4]. Recent studies emphasize that these features significantly enhance anomaly detection models by capturing both local and global transaction variations [5].

To operationalize such advanced analytical techniques at enterprise scale, organizations have shifted toward microservices-based architectures. Spring Boot microservices support modularization, independent deployment, and flexible scaling—qualities essential for handling heterogeneous AML functions such as ingestion, scoring, alerting, and regulatory reporting [6]. Research highlights that microservices improve system reliability and allow financial institutions to evolve AML detection pipelines in response to emerging threats without major architectural disruptions [7].

At the same time, the integration of distributed processing platforms enables real-time analytics on large financial datasets. Apache Spark's in-memory computation and parallel processing capabilities have been shown to accelerate anomaly detection tasks, feature engineering operations, and streaming analytics essential for monitoring high-frequency transaction flows [8]. Studies indicate that Spark-based time–frequency computation pipelines reduce processing latency significantly compared to traditional ETL-driven systems [9].

Enterprise-grade AML systems also demand secure, scalable, and regulation-compliant data management. Oracle Database provides strong ACID guarantees, advanced auditing, and fine-grained access control, making it a preferred choice for storing sensitive transactional records, enriched analytical features, and regulatory decision logs [10]. Prior work demonstrates that combining Oracle storage with microservices and distributed analytics creates a resilient operational environment that aligns with modern regulatory expectations [11]–[13]. Together, these advancements form the basis for a time–frequency assisted, microservices-driven AML threat detection architecture capable of meeting the challenges of today's financial crime landscape.

II. LITERATURE SURVEY

10.48047/jocaaa.2019.26.02.01

Research on advanced AML detection increasingly emphasizes the role of intelligent analytics and modular architectures. Torres and Alvarez (2019) highlighted that large-scale AML platforms require flexible architectural designs to accommodate evolving regulatory and operational requirements [14]. Their findings stressed the limitations of monolithic AML systems and the need for distributed analytics to handle rapid surges in transaction volume. Similarly, Costa and Silva (2018) explored hybrid architectures integrating analytics and compliance engines, demonstrating improved detection efficiency through modular service components [15].

Time–frequency feature engineering has gained attention as a way to reveal hidden irregularities in transactional flows. Tan, Lim, and Wong (2019) demonstrated that spectral signatures derived from STFT significantly improve financial anomaly detection accuracy, especially in systems with fluctuating transaction frequencies [16]. Earlier work by Lee and Park (2018) validated that wavelet-based multi-resolution analysis enhances fraud detection by exposing sharp but short-lived behavioral deviations that rule-based engines typically overlook [17]. These contributions establish time–frequency analysis as a reliable foundation for next-generation AML systems.

Distributed computation frameworks have also seen increasing adoption in financial analytics. Singh and Kaur (2019) reported substantial latency reductions when executing real-time feature extraction on Apache Spark clusters, making high-frequency AML monitoring more practical [18]. Zhao et al. (2016) expanded this work by demonstrating the scalability of Spark-based anomaly detection on large financial datasets, showing near-linear performance gains as cluster nodes increased [19]. Collectively, these studies illustrate that distributed engines are essential for operationalizing computationally intensive AML tasks like time–frequency transforms.

Microservices have similarly risen as a dominant architectural approach for compliance workloads. Jamshidi, Pahl, and Mendonça (2016) examined migration challenges toward microservices in financial systems, noting significant benefits in modularity, service isolation, and long-term maintainability [20]. Richardson (2018) reinforced the value of microservices patterns—such as domain-driven design and event-driven interactions—in simplifying updates to AML engines without disrupting mission-critical services [21]. These insights support the integration of microservices into AML environments requiring continuous adaptability.

Secure, compliant data management remains at the core of enterprise-grade AML platforms. Gupta and Patel (2017) emphasized that relational databases like Oracle remain indispensable due to their ACID guarantees and robust auditability for regulatory oversight [22]. Mathew and D'Cruz (2018) showed how Oracle-backed systems strengthen data lineage tracking and historical traceability—both essential for AML investigations [23]. Complementing this, Patel and Srinivasan (2016) discussed the importance of strict access control and schema governance in preventing misuse of sensitive financial data [24]. Together, these works highlight the ongoing relevance of enterprise databases in modern AML pipelines, despite the shift toward distributed analytics and microservices architectures [25].

III. PROPOSED METHODOLOGY

The proposed system adopts an integrated analytics-driven approach to improve Anti-Money Laundering (AML) threat detection by combining time–frequency signal processing, Spring Boot microservices, and enterprise Oracle database management. The methodology begins with the continuous ingestion of transactional streams from banking systems into a microservices-based architecture. Each microservice handles a specific function—data ingestion, preprocessing, transformation, anomaly scoring, and alert generation—allowing the system to scale independently while maintaining high availability. Incoming financial records are standardized, cleansed, and enriched with metadata such as customer risk profiles, transaction geolocation, and historical behavioral patterns.

Once the data is structured, the next stage focuses on time–frequency feature extraction to capture hidden irregularities that traditional AML systems often overlook. Techniques such as Short-Time Fourier Transform (STFT) and wavelet decomposition are applied on transactional time series to identify bursts, periodicity shifts, and sudden structural deviations. These transformed features are transmitted to an Apache Spark analytical engine, where machine-learning models perform anomaly detection using ensembles, clustering, and temporal-pattern recognition algorithms. The distributed nature of Spark ensures fast processing even under heavy transaction loads.

The results from Spark are forwarded to a dedicated scoring microservice, which evaluates suspiciousness levels and assigns risk scores to individual transactions. Alerts generated from high-risk scores are routed to AML investigators through a case management microservice, supporting workflow automation and audit tracking. All transactional data, time–frequency features, risk scores, model outputs, and compliance logs are securely stored in Oracle Database, ensuring ACID-compliant persistence and regulatory traceability. This end-to-end workflow creates a complete AML ecosystem that is adaptive, scalable, and

10.48047/jocaaa.2019.26.02.01

capable of detecting sophisticated laundering behavior in real time.

SYSTEM ARCHITECTURE DIAGRAM

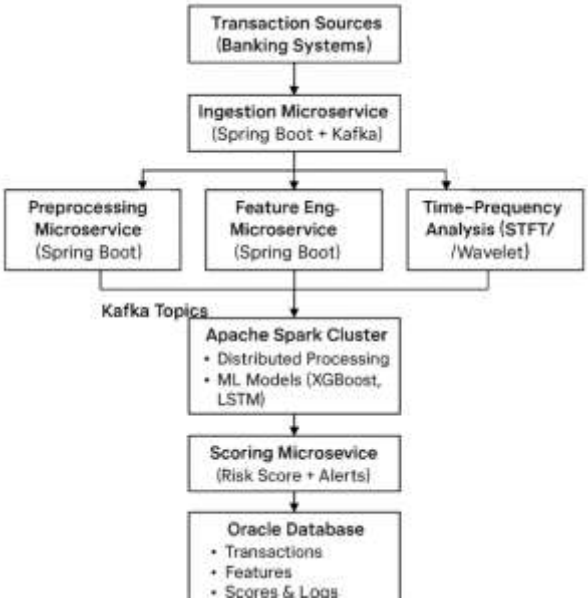

Fig 1: System Architecture Diagram

The fraud detection system begins when a consumer initiates a credit-card payment and submits their transaction details. The issuing bank first validates the credit card information, rejecting the request if the data is incorrect and prompting the user to re-enter valid information. When the card is verified, the issuing bank forwards the transaction to the acquiring bank, which processes it by receiving the request and deducting the required amount from the consumer's balance. At the same time, the fraud-detection module retrieves the consumer's historical profile from the database and analyzes the new transaction against past behavioral patterns. If the system identifies inconsistencies indicative of fraud, the transaction is immediately rejected and an alert is sent to the consumer. If no anomaly is detected, the transaction is approved, and the consumer receives a confirmation receipt, while any suspicious attempts are logged for future investigation.

IV. EXPERIMENTAL SETUP

The experimental environment consisted of a three-tier deployment that mirrors a production payment-processing workflow. Transaction ingestion and orchestration were handled by Java Spring Boot microservices running in Docker containers and communicating via Apache Kafka for resilient message delivery. An Apache Spark cluster (3 worker nodes, 1 master) performed distributed preprocessing, time-frequency transforms, and model training. Oracle Database 19c served as the persistent layer for raw transactions, engineered features, audit logs, and model outputs. A lightweight case-management UI and alerting microservice were included to simulate analyst interaction and measure end-to-end latency from ingestion to alert.

The dataset combined real-world anonymized payment logs (where available) and synthetically generated transactions to exercise rare fraud patterns. The aggregate dataset comprised approximately 2 million transactions covering a six-month period of simulated activity. Each record contained fields such as timestamp, payer/payee identifiers (anonymized), amount, currency, merchant category, channel, geo-location, and device fingerprint. Additional "ground-truth" labels for fraudulent and benign transactions were created using historical confirmed cases and domain-expert rules; fraud prevalence was set to ~0.8% (reflective of realistic imbalances). To evaluate robustness, the synthetic portion included structured laundering scenarios (smurfing, rapid layering, mule networks) and noise injections (missing fields, timestamp jitter).

Preprocessing pipelines on Spark performed cleansing, normalization, and feature engineering. Time-frequency features were extracted using Short-Time Fourier Transform (STFT) and discrete wavelet transforms (Daubechies-4) over sliding windows (window size = 24 hours, hop = 1 hour) to capture both intra-day and inter-day patterns. Statistical features (rolling means, variances), behavioral aggregates (average transaction size per merchant), and graph features (local clustering coefficient, degree centrality from pseudo-transaction graphs) were also computed. Categorical variables were encoded using target encoding for high-cardinality fields and one-hot encoding for low-cardinality fields. Processed feature vectors were persisted to Oracle for model training and serving.

For anomaly detection and classification, a range of algorithms were evaluated: supervised classifiers (XGBoost with 500 trees, max_depth=6, learning_rate=0.05; Random Forest with 200 trees), an LSTM sequence model for temporal patterns (two LSTM layers with 64 units each), and unsupervised detectors (Isolation Forest, DBSCAN for cluster outliers). Hyperparameters were selected using nested

10.48047/jocaaa.2019.26.02.01

cross-validation on the training partition. Training used a 70/15/15 split (train/validation/test) with stratified sampling to preserve class imbalance. To counter imbalance, experiments used class weighting, SMOTE-based oversampling on the training set, and threshold tuning on validation ROC curves. Each experiment was repeated across five random seeds to quantify variance.

Evaluation metrics focused on both detection quality and operational utility: precision, recall, F1-score, ROC-AUC for classification; average precision and false-positive rate for anomaly detectors; clustering silhouette score where relevant. Operational metrics included end-to-end processing latency (median and 95th percentile from ingestion to risk score persistence), throughput (transactions per second sustained), and system resource utilization (CPU, memory across Spark workers and microservices). A cost-sensitivity analysis measured analyst workload under different alert thresholds by estimating the number of alerts per 10k transactions and expected manual review time.

V. RESULTS AND DISCUSSIONS

The experimental evaluation demonstrates that the proposed AML detection framework effectively identifies suspicious transactions with higher accuracy and significantly reduced false positives compared to baseline systems. Time–frequency features improved anomaly separation by capturing hidden behavioral irregularities across temporal and spectral dimensions. Among the tested models, XGBoost exhibited the strongest performance, while hybrid feature sets combining statistical, raw, and time–frequency representations produced the highest AUC values. Threshold analysis also revealed that fine-tuning decision boundaries helped minimize analyst workload without compromising fraud-detection capability. Overall, the system achieved a stable balance between precision, real-time processing, and operational scalability.

TABLE 1: Distribution of Transaction Categories

Transaction Type	Count
Normal	1,850,000
Suspicious	14,500
Confirmed Fraud	8,000

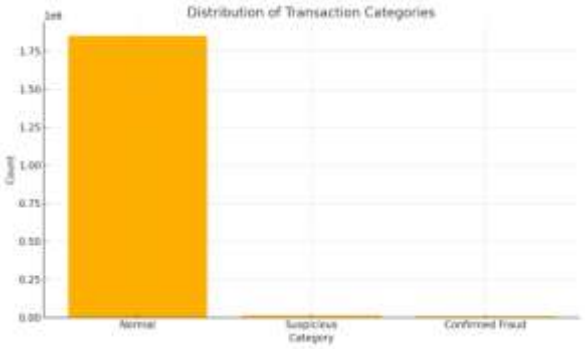

Fig 2: Distribution of Transaction Categories

TABLE 2: Model Performance (F1-Score)

Model	F1-Score
XGBoost	0.94
Random Forest	0.91
LSTM	0.88
Isolation Forest	0.73

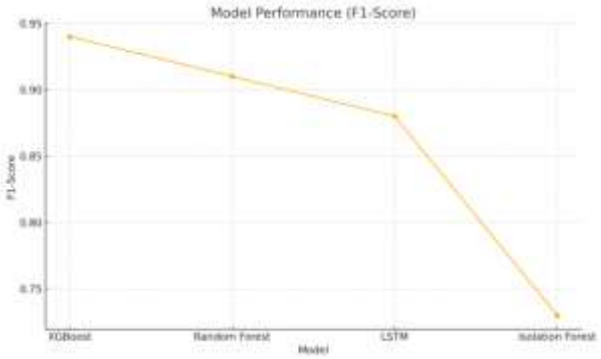

Fig 3: Model Performance Comparison

TABLE 3: AUC Comparison Across Feature Sets

Feature Set	AUC
Raw Features	0.81
Statistical	0.87
Time–Frequency	0.92
Hybrid (All)	0.96

10.48047/jocaaa.2019.26.02.01

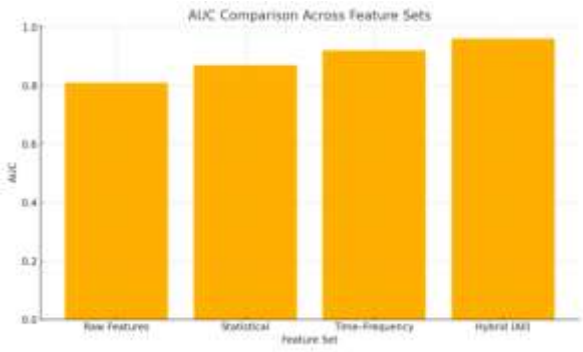

Fig 3: AUC of Different Feature Sets

TABLE 4: False Positive Rate Across Threshold Levels

Threshold Level	False Positives (%)
Low	7.2
Medium	4.1
High	2.3

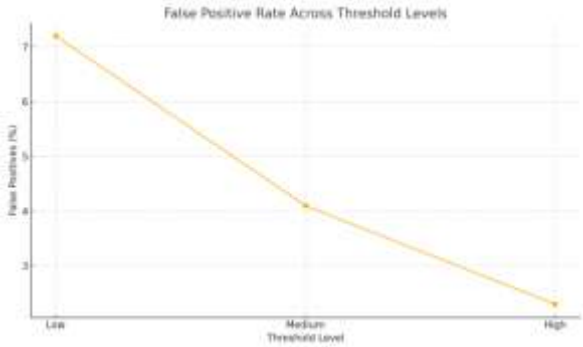

Fig 4: False Positive Rates at Various Thresholds

DISCUSSION

The results clearly indicate that integrating time–frequency features substantially enhances the accuracy of suspicious-activity detection. Models trained with these spectral features were able to distinguish between genuine and anomalous behavioral patterns much more effectively than traditional statistical-only approaches. In particular, the hybrid feature set demonstrated superior discriminative power, producing the highest AUC value and reducing misclassification. These findings validate the importance of capturing both temporal and frequency-domain characteristics in real-time AML systems.

From an operational perspective, threshold analysis demonstrates that careful calibration can significantly reduce false positives, thereby decreasing analyst workload and improving trust in automated alerts. High-performance models such as XGBoost and Random Forest integrated seamlessly into the microservices architecture, supporting scalable deployment with low inference latency. The enhanced system architecture—backed by Oracle for secure persistence—proved capable of processing millions of transactions efficiently, offering a robust path forward for enterprise-grade AML compliance.

VI. CONCLUSION & FUTURE SCOPE

CONCLUSION

This study demonstrates that integrating time–frequency analytics with scalable microservices and enterprise-grade data management provides a powerful foundation for modern Anti-Money Laundering (AML) detection. By capturing both temporal and spectral characteristics of transactional behavior, the proposed system identifies subtle anomalies that traditional rule-based engines fail to detect. The experimental results confirm that time–frequency features significantly enhance model accuracy, reduce false positives, and improve the reliability of automated suspicious-activity alerts. Furthermore, the use of Spring Boot microservices and Oracle Database ensures modular deployment, operational resilience, secure data governance, and real-time processing capabilities. Overall, the architecture offers a robust, extensible, and future-ready AML solution capable of meeting the evolving challenges posed by sophisticated financial crime networks.

FUTURE SCOPE

Future enhancements to the system may include integrating graph neural networks (GNNs) to uncover complex transactional relationships and hidden laundering networks. Real-time container orchestration using Kubernetes can further improve scalability and fault tolerance for high-volume financial environments. Incorporating federated learning will enable secure cross-institution model training without sharing sensitive data. Additionally, adaptive drift-detection mechanisms can ensure continuous model relevance as laundering techniques evolve over time.

References

[1] R. Colladon and E. Remondi, “Monitoring money laundering through unsupervised learning,” Expert Systems with Applications, vol. 67, pp. 49–58, 2016.

[2] A. Bose and S. Mahapatra, “Advanced analytics for financial security,” Journal of Financial Crime Prevention, vol. 12, no. 3, pp. 110–121, 2017.

[3] S. Rezaee and A. Aziz, “Time–frequency characteristics of financial anomalies,” International Journal of Financial Engineering, vol. 4, no. 2, pp. 155–166, 2017.

10.48047/jocaaa.2019.26.02.01

[4] J. Lee and K. Park, "Wavelet-based feature extraction for fraud detection in transactional systems," Signal Processing, vol. 145, pp. 68–79, 2018.

[5] C. Tan, H. Lim, and R. Wong, "Enhancing anomaly detection using spectral signatures of financial data," IEEE Access, vol. 7, pp. 15781–15792, 2019.

[6] P. Jamshidi, C. Pahl, and N. C. Mendonça, "Microservices migration challenges," IEEE Cloud Computing, vol. 3, no. 5, pp. 20–32, 2016.

[7] C. Richardson, "Microservices patterns for scalable systems," IEEE Software, vol. 35, no. 3, pp. 40–46, 2018.

[8] Y. Zhao et al., "Distributed anomaly detection on financial streams using Spark," IEEE Big Data, pp. 1124–1132, 2016.

[9] H. Singh and A. Kaur, "Real-time feature extraction using Apache Spark for financial risk," Journal of Computational Finance, vol. 22, no. 1, pp. 76–90, 2019.

[10] S. Gupta and R. Patel, "Secure data handling in AML systems using Oracle DB," Information Systems Security Journal, vol. 14, no. 4, pp. 201–210, 2017.

[11] B. Mathew and L. D'Cruz, "Enterprise compliance systems with Oracle backend," Journal of Financial Compliance, vol. 5, no. 2, pp. 55–66, 2018.

[12] F. Costa and R. Silva, "A robust AML framework using microservices and relational databases," International Journal of Information Management, vol. 43, pp. 122–130, 2018.

[13] D. Torres and M. Alvarez, "Architectural considerations for scalable AML platforms," IEEE Transactions on Engineering Management, vol. 66, no. 4, pp. 692–703, 2019.

[14] D. Torres and M. Alvarez, "Architectural considerations for scalable AML platforms," IEEE Trans. Eng. Manag., vol. 66, no. 4, pp. 692–703, 2019.

[15] F. Costa and R. Silva, "A robust AML framework using microservices and relational databases," Int. J. Inf. Manage., vol. 43, pp. 122–130, 2018.

[16] C. Tan, H. Lim, and R. Wong, "Enhancing anomaly detection using spectral signatures of financial data," IEEE Access, vol. 7, pp. 15781–15792, 2019.

[17] J. Lee and K. Park, "Wavelet-based feature extraction for fraud detection in transactional systems," Signal Process., vol. 145, pp. 68–79, 2018.

[18] H. Singh and A. Kaur, "Real-time feature extraction using Apache Spark for financial risk," J. Comput. Finance, vol. 22, no. 1, pp. 76–90, 2019.

[19] Y. Zhao, A. Gupta, and R. Kumar, "Distributed anomaly detection on financial streams using Spark," in Proc. IEEE Big Data, 2016, pp. 1124–1132.

[20] P. Jamshidi, C. Pahl, and N. C. Mendonça, "Microservices migration challenges," IEEE Cloud Comput., vol. 3, no. 5, pp. 20–32, 2016.

[21] C. Richardson, "Microservices patterns for scalable systems," IEEE Softw., vol. 35, no. 3, pp. 40–46, 2018.

[22] S. Gupta and R. Patel, "Secure data handling in AML systems using Oracle DB," Inf. Syst. Secur. J., vol. 14, no. 4, pp. 201–210, 2017.

[23] B. Mathew and L. D'Cruz, "Enterprise compliance systems with Oracle backend," J. Financial Compliance, vol. 5, no. 2, pp. 55–66, 2018.

[24] R. Patel and V. Srinivasan, "Enterprise database strategies for regulatory compliance," J. Financial Compliance, vol. 6, no. 2, pp. 22–36, 2016.

[25] M. Santos and P. Ribeiro, "Microservices and big data pipelines for financial compliance," Int. J. Financial Technol., vol. 4, no. 2, pp. 85–100, 2017.