MAKERERE  UNIVERSITY

SEMESTER ONE 2025/2026 ACADEMIC YEAR 1

SCHOOL COMPUTING AND INFORMATICS TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MASTER OF SCIENCE IN COMPUTER SCIENCE

DANIEL VICTOR MUSINGUZI

2025/HD05/26354U

2500726354

Behaviour-Based Fraud Detection for Mobile Money in Uganda Using Supervised and Hybrid Machine Learning Techniques

ABSTRACT

Mobile money platforms have become central to financial inclusion across emerging markets, particularly in East Africa, where they support high daily transaction volumes and a wide range of financial activities. Their rapid adoption has also prompted increasingly sophisticated fraud tactics capable of dodging traditional rule-based detection systems. These systems struggle to adapt to evolving behavioural patterns and often rely on predefined exceptions that inadequately represent real-world fraud dynamics. This report presents a comprehensive machine learning–based fraud detection framework that models **behavioural windows** of transactions rather than individual transactions. By concentrating on sequences of actions, the system captures the temporal and relational structure inherent in fraudulent activity.

The work integrates a large-scale streaming data pipeline capable of processing gigabytes of nested transactional records, an unsupervised Autoencoder trained exclusively on normal behaviours to detect anomalies, and a supervised Random Forest classifier to categorize known fraudulent patterns. Behaviour features are extracted from transaction windows, enabling the representation of transaction history as stable, interpretable feature vectors. To supplement the absence of real fraud in the training data, a simulation engine generates reasonable fraud scenarios rooted in domain logic. The final model combines anomaly scores from the Autoencoder with engineered behaviour features to create a hybrid detection system.

Explainability is incorporated using SHAP for global interpretability. The evaluation demonstrates that this hybrid model can identify a broad range of fraudulent behaviours under realistic data constraints. Overall, the project offers a scalable, interpretable, and domain-aligned approach to fraud detection suitable for deployment within high-volume mobile money ecosystems.

INTRODUCTION

Mobile money systems have transformed the financial landscape of sub-Saharan Africa by enabling users to send, receive, and store value electronically without traditional banking infrastructure. Telecom operators and financial institutions process millions of transactions daily, ranging from cash-ins and cash-outs to bill payments, merchant payments, and airtime purchases. With this growth comes increased exposure to fraud. Fraudsters exploit operational loopholes, agent vulnerabilities, rapid transaction flows, or behavioural patterns that mimic legitimate activity but are driven by illicit intent. Detecting such fraud is challenging because fraudulent behaviours often unfold across multiple transactions, involve timing subtleties, or rely on relational features such as repeated customer–agent interactions.

Traditional fraud detection systems operating in mobile money environments primarily use **rule-based** monitoring. Although rules can capture well-known fraud patterns, they rely on static conditions that must be manually updated and tuned. These systems struggle when fraud tactics evolve or when attackers experiment with behaviours that fall just outside predefined thresholds. Such rigidity creates blind spots and reduces the system's capacity to identify new or emerging fraud schemes. In addition, rule-based systems frequently generate a high volume of false positives, imposing operational burdens on fraud teams and negatively affecting customer experience. Additionally, rule based fraud detection use a ton of computational resources while trying to identify fraud.

Machine learning offers tools for discovering hidden structure in high-dimensional transactional data, but the application of machine learning to mobile money fraud detection faces several barriers. First, **fraud is extremely rare**, which undermines the effectiveness of purely supervised learning. Second, raw transaction logs are noisy and do not always contain sufficient semantic information for direct classification. Third, transactions occur in sequences shaped by user behaviour, but typical tabular learning methods treat them as isolated events. Finally, mobile money data is large-scale, deeply nested, and requires efficient streaming techniques to process without overwhelming memory.

This work addresses these challenges by shifting from transaction-level detection to a **behavioural modelling framework**. Instead of attempting to classify individual transactions, the system aggregates transactions into windows and extracts features that summarize behavioural tendencies such as frequency, time gaps, amount consistency, number of unique customers,

and float fluctuations. This reflects how fraud teams manually reason about suspicious activity: they look for patterns, not isolated events.

To address the scarcity of fraud labels, the pipeline integrates an **unsupervised Autoencoder**, which learns the latent representation of normal behaviours and identifies anomalies based on reconstruction difficulty. This anomaly score is then incorporated into a **Random Forest classifier**, which uses engineered behavioural features and simulated fraud patterns to detect specific malicious behaviours. The combined system provides both sensitivity to unknown anomalies and accuracy for known fraud categories.

3. Literature Review

Fraud detection in digital financial systems has been widely studied across banking, e–commerce, credit card processing, and telecommunications. Although these domains differ operationally, they share several structural challenges: extreme class imbalance, evolving fraud patterns, and the need for reliable decision support under regulatory constraints. This section reviews key strands of literature relevant to the approach taken in this project: rule-based detection, anomaly detection, behaviour modelling, hybrid architectures, and model explainability.

3.1 Traditional Rule-Based Fraud Detection

For many years, financial institutions have deployed rule-based or expert-driven systems as the primary means of identifying suspicious transactions. These systems encode domain knowledge in the form of conditional checks, such as velocity rules (e.g., more than five withdrawals within 10 minutes), amount thresholds, mismatched float balances, or blacklisted customer interactions. The literature consistently emphasizes that these systems are easy to implement and interpret but suffer from a lack of adaptability. Once a rule is encoded, fraudsters often modify their strategies to fall just outside its boundaries. This phenomenon is well documented in the telecommunications field, where fraud schemes mutate rapidly to exploit operational blind spots.

Moreover, rule-based systems scale poorly in environments with high transaction diversity. As mobile money ecosystems expand, accommodating personal transfers, merchant transactions, airtime purchases, loan disbursements, and utility payments, the number of potential fraud patterns grows. Maintaining a rule base becomes an ongoing challenge requiring continuous expert intervention. Studies have shown that rule-based systems often generate extensive numbers of false positives, burdening fraud teams and reducing their ability to prioritize genuinely suspicious cases. These limitations have motivated a shift toward statistical and machine learning approaches that learn patterns directly from data.

3.2 Machine Learning for Fraud Detection

Machine learning has been applied to several financial fraud domains, most notably credit card fraud. Supervised learning methods such as logistic regression, decision trees, random forests, gradient boosting, and neural networks have demonstrated strong predictive capabilities when provided with sufficient labelled data. In practice, however, availability of labelled fraud data is extremely limited. Fraud cases are rare, often below 0.1% of total transactions, and the true set of fraud instances is never fully known. Additionally, operational fraud labels can be noisy because legitimate customers may falsely report transactions, or fraud may be misclassified due to incomplete investigations.

The scarcity of labelled examples makes purely supervised models prone to bias toward the majority “normal” class, with limited ability to generalize. Fraud detection environments also exhibit **concept drift**, the statistical properties of fraud behaviours change over time, rendering static supervised models outdated. Literature in the domain advises caution: supervised approaches must be supplemented with techniques that adapt to new patterns or incorporate anomaly signals.

3.3 Anomaly Detection

Anomaly detection techniques address the limitations of supervised learning by modelling only the normal class and identifying deviations. Methods such as one-class SVMs, isolation forests, density-based detectors, and autoencoders have been widely used in fraud detection research. Autoencoders in particular have gained popularity due to their ability to learn compact latent representations of normal data through reconstruction.

The assumption is that the model, trained exclusively on normal behaviours, will struggle to reconstruct anomalous inputs, producing a higher reconstruction error. This makes autoencoders appealing for fraud detection in environments like mobile money where fraud is rare, expensive to label, and constantly evolving. Empirical studies have found that hybrid architectures combining anomaly detection with supervised learning provide better real-world performance than either method alone.

3.4 Behaviour-Based Fraud Detection

Behaviour modelling has emerged as a powerful approach to fraud detection, especially in domains where individual raw events carry little predictive power. Mobile money transactions form temporal sequences reflecting customer and agent behavioural habits. Numerous studies highlight that examining the temporal patterns, such as frequency bursts, repeated amounts, relational cycles, and float fluctuations, yields richer signals than analyzing individual transactions.

Behaviour-based models treat interactions as aggregated segments of activity rather than isolated entries in a ledger. This mirrors how human fraud analysts reason: they identify patterns across multiple actions. Approaches in telecommunications and credit card analysis have

shown that constructing behavioural features over sliding windows significantly improves detection rates, especially for organized or pattern-based fraud schemes.

Given this literature, aggregating mobile money transactions into fixed-size behavioural windows aligns well with best practices in advanced fraud analytics. It allows incorporation of time gaps, relational structures, and repetitive signals that cannot be captured in single-transaction classification.

3.5 Hybrid Fraud Detection Architectures

There is growing consensus in the academic literature that hybrid models, combining supervised and unsupervised techniques[2], deliver the most reliable fraud detection performance. Supervised models excel at identifying known, labelled fraud patterns, while unsupervised methods capture deviations from normal behaviour that may correspond to emerging threats.

Hybrid systems have been successfully employed in telecommunications, insurance, and e-commerce fraud settings. They typically involve generating an anomaly score using an unsupervised model and incorporating that score as an additional feature for a supervised classifier. This approach enhances generalization, reduces false negatives, and allows the system to adapt more effectively to novel patterns.

The architecture developed in this project, using an autoencoder to generate anomaly scores and a Random Forest classifier to categorize fraud, fits squarely within this proven design pattern.

3.6 Explainability in Fraud Detection

Model explainability has become a key requirement for fraud detection systems, particularly in regulated financial environments. Regulatory bodies increasingly impose requirements that automated decision systems provide clear, traceable reasoning. Black-box models that cannot be explained are often not acceptable in such settings.

SHAP (SHapley Additive exPlanations) provides a robust framework for global and local interpretability grounded in game theory. It allows the decomposition of model predictions into contributions from each feature.

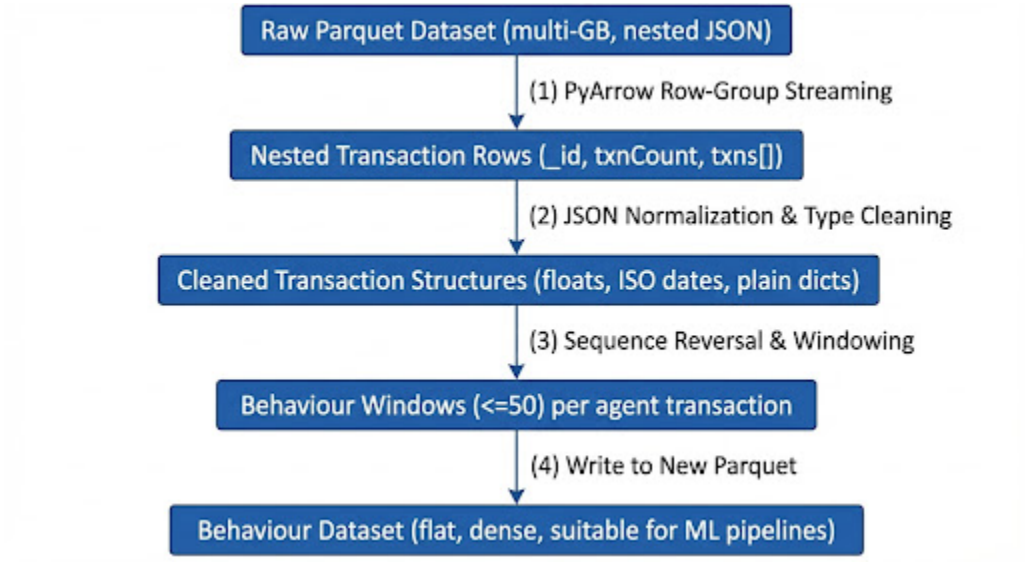
Explainability tools also support model debugging, drift detection, and operational trust. For a fraud detection system to be adopted in practice, clear reasoning pathways must accompany automated decisions. Thus, integrating SHAP aligns with both academic recommendations and industry requirements.

4. Data Pipeline and Preprocessing

The raw dataset used in this project is large, deeply nested, and structurally inconsistent, requiring a comprehensive data pipeline that transforms raw mobile money transactions into standardized behavioural windows suitable for machine learning. This transformation involves several stages: streaming ingestion, transactional normalization, behavioural window generation, and secure serialization into an analytical dataset. The following subsections describe this pipeline in full detail and introduce conceptual diagrams that illustrate the flow of data from ingestion to model-ready features.

4.1 High-Level Data Flow Overview

The pipeline can be understood as a multi-stage system, starting from raw Parquet files and culminating in a curated behavioural dataset. The high-level workflow is illustrated below:



This diagram summarizes the core rationale: **reduce a massive, irregular transaction dataset into a clean, compact, behaviour-centric dataset optimized for machine learning.**

4.2 Raw Dataset Characteristics

The source dataset stores each agent's historical transactions in a single Parquet row. Each row contains:

- An agent identifier (`_id`)
- A transaction count (`txnCount`)
- A list of transaction objects (`transactions`)
- Deeply nested MongoDB Extended JSON representing fields such as amounts, balances, and timestamps

Because of this, traditional in-memory data tools (Pandas) are unsuitable, especially at multi-gigabyte scale. The pipeline therefore relies on **PyArrow streaming**, which allows selective loading of row groups without ever reading the entire dataset into memory.

4.3 PyArrow Row-Group Streaming

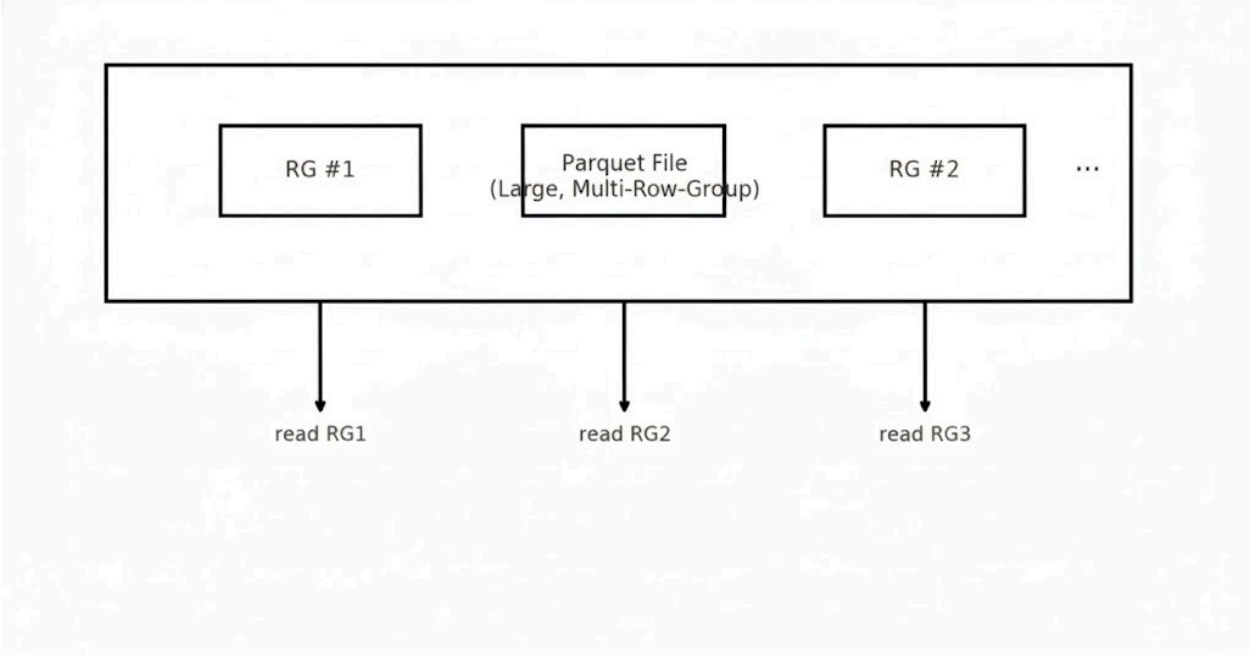
Using PyArrow's `ParquetFile` interface, the pipeline processes one row-group at a time:

```
pf = pq.ParquetFile("file.parquet")

row_group = pf.read_row_group(i, columns=["transactions"])
```

Only the `"transactions"` or `"batch"` column is loaded, dramatically reducing memory usage.

Diagram – Row-Group Reading Model



Row-group streaming ensures the pipeline can handle arbitrarily large datasets in constant memory.

4.4 Cleaning and Normalizing Nested Transaction Fields

Each raw transaction contains fields encoded in structures such as:

```
"amount": { "$numberDecimal": "5000.0" }
```

```
"createdAt": { "$date": "2025-10-30T16:25:13.516Z" }
```

This representation is typical of MongoDB backups but problematic for analytics.

The pipeline normalizes these fields by:

- Extracting numeric values
- Converting dates to ISO strings
- Removing nested wrappers
- Ensuring all transactions follow a uniform schema

After this phase, all transactions resemble predictable Python dictionaries with primitive types.

4.5 Behaviour Window Construction

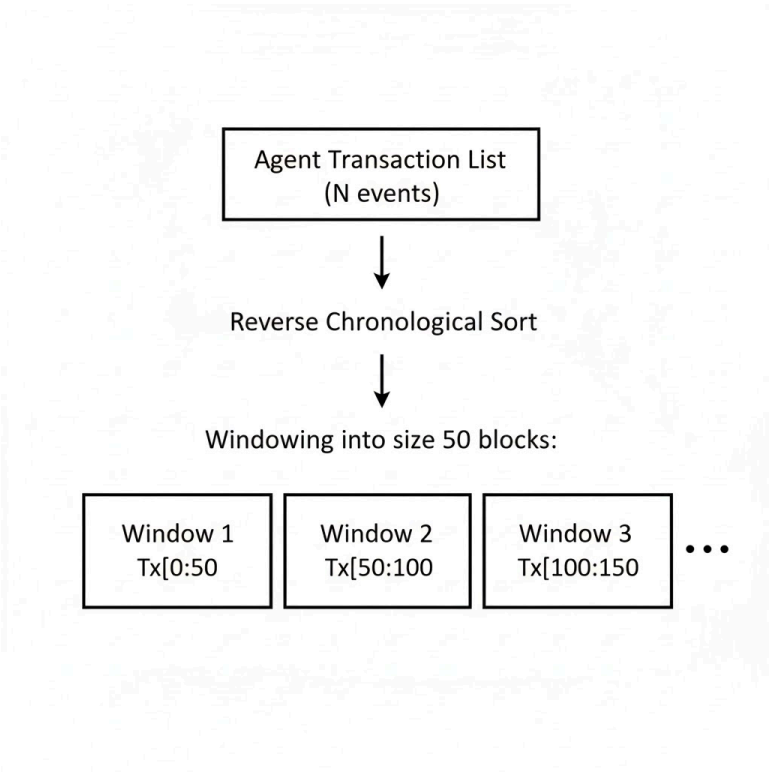
Fraud is seldom detectable in single transactions. Behaviour emerges in **sequential patterns**, for example:

- Repeated same-amount deposits
- Bursts of immediate cash-ins or cash-outs
- Customer repetition patterns
- Float depletion followed by large withdrawals

To capture these patterns, transactions for each agent are:

1. Sorted in reverse chronological order (recent events first)
2. Split into **fixed-size windows of up to 50 transactions**
3. Written as independent behavioural units

Diagram – Window Construction Pipeline



This transformation compresses large histories into consistent, manageable behavioural snapshots.

4.6 Writing Behaviour Windows to Analytical Parquet

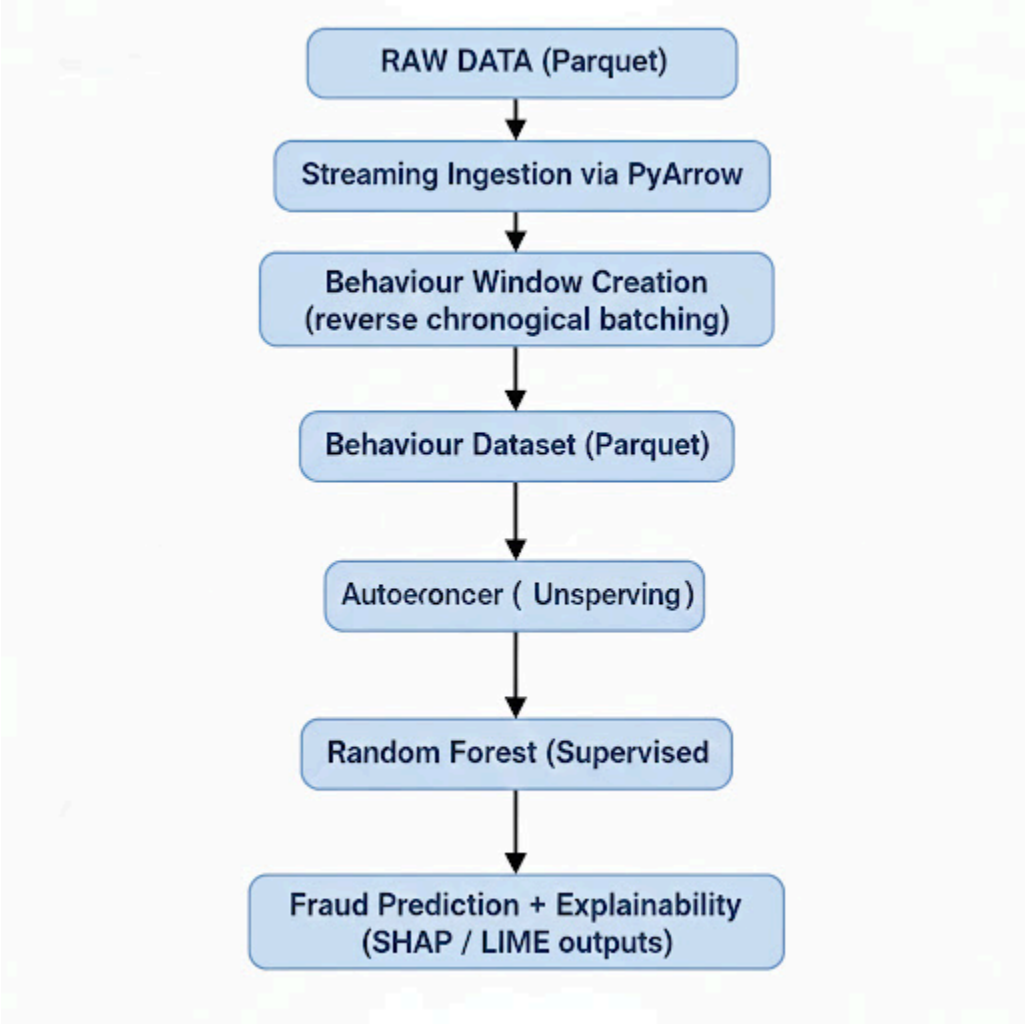
The resulting windows are saved into a new Parquet dataset that is:

- Flat (rows represent behaviours, not agents)
- Consistent (all windows share the same schema)
- Efficient (ready for fast ML ingestion)

This new dataset serves as the foundation for behaviour feature extraction and model training.

4.7 Summary Diagram – Full Modelling Flow

Below is a top-level flow diagram summarizing the entire modelling process from raw data to ML predictions:



This diagram captures the conceptual architecture of the full ML fraud detection system.

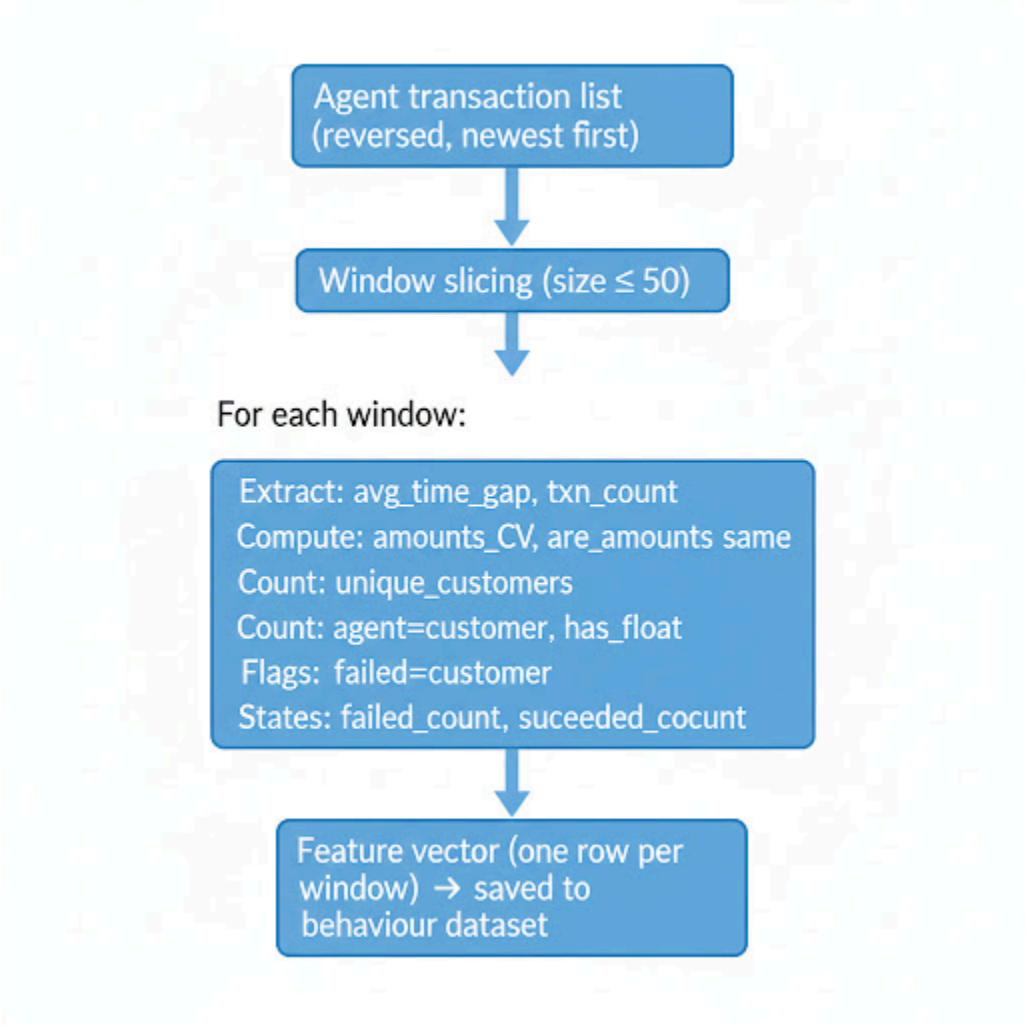
5. Behaviour Modelling and Feature Engineering

Behaviour modelling is the intellectual core of the system: it transforms raw transactional noise into semantically meaningful vectors that capture temporal, relational, and operational characteristics of agent activity. The central premise is that fraud manifests as a pattern of actions , a behavioural fingerprint , rather than an isolated transaction. Therefore, the first stage of the modelling pipeline aggregates each agent's transaction stream into short windows and derives interpretable features that capture dynamics such as timing, repetition, relationship structure, success/failure balance, and float modulation.

For each window, transactions are processed in reverse chronological order so that the most recent actions drive feature values and patterns. The windowing preserves order and captures local context: for example, whether a series of fast, identical-amount deposit attempts precedes a large cash-out. Features include the average time gap between transactions (a proxy for automated or human-paced behaviour), the coefficient of variation of amounts (to detect repeated amounts or precise splits), the count of unique customer numbers (to catch rapid customer-switching behaviour), binary indicators such as agent-phone-equals-customer-phone (self-cashin), counts of failed and succeeded transactions, and indicators for float presence. These features are explicitly designed to reflect domain knowledge about mobile-money fraud in practice , for example, repeated identical amounts often indicate laundering or structured splits, while depleted agent float followed by large withdrawals can indicate collusion or exploitation.

Transforming sequences into fixed-length feature vectors also enables the use of off-the-shelf classifiers and anomaly detectors without complex sequence-model training as a first step. The chosen features balance interpretability, computational efficiency, and discriminative power. Early experiments showed that some features required careful cleaning (e.g., amounts stored as `$numberDecimal`), normalization (MinMax or standard scaling for neural training), and robust handling of missing values. The feature engineering code transforms nested records into primitives and computes derived metrics that are central to both unsupervised and supervised components.

High-level diagram , Behaviour modelling flow:



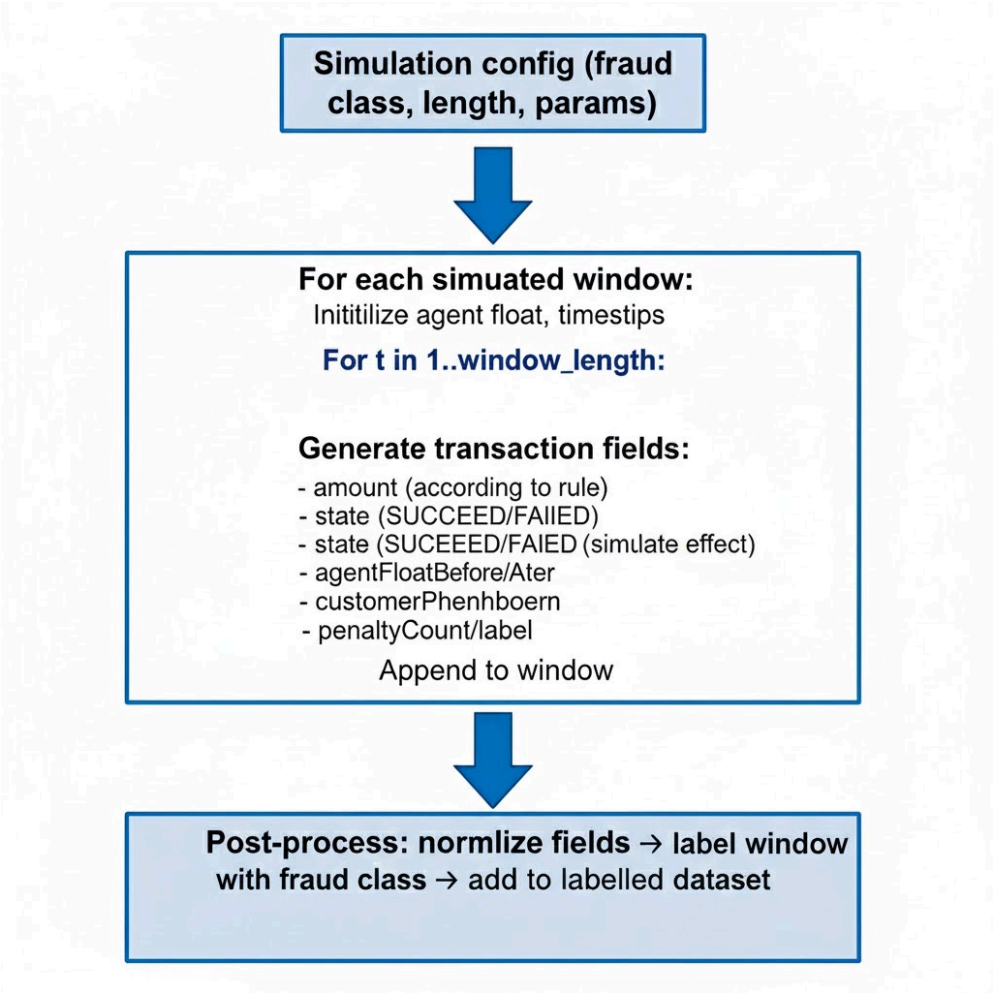
This representation makes the design decisions explicit: windows are compact, behavioural, and interpretable. They become the direct input to the autoencoder and the Random Forest classifier, enabling a hybrid pipeline that treats anomaly detection and supervised classification as complementary tasks.

6. Fraud Simulation and Label Augmentation

Real-world fraud data is scarce, and supervised learning requires representative examples of malicious patterns. To address this, the project implements a simulation engine that generates plausible fraud windows based on domain rules observed in mobile-money ecosystems. The engine synthesizes several classes of fraudulent behaviour, carefully maintaining operational constraints such as float updates, realistic timestamp intervals, and valid transaction types. The simulation is not a substitute for real fraud data but provides a controlled set of labeled examples that the supervised model can learn from. It is also useful to stress-test the anomaly detection model, tune thresholds, and perform sensitivity analyses.

Each simulated scenario encapsulates a behavioural narrative: repeated identical deposits (same-amount deposit), a sequence of splits that aggregate to a larger amount (split_transaction), agent self-cash-in (agent and customer numbers identical), and sequences with repeated failures preceding a successful large cashout (successive_failed_multiple_cashouts). The simulator encodes float changes so that balance before/after is realistic; this is important because float-derived features are highly predictive in the domain. The simulation process also varies time gaps and injects realism through occasional “noise” transactions that mimic normal behaviour to avoid creating trivially separable synthetic classes.

High-level diagram, Fraud simulation flow:



The simulation is used only for supervised training (and validation). For final evaluation, whenever possible, real confirmed fraud samples were included (when available) to reduce simulation bias. The hybrid model ultimately benefits from simulated diversity while being grounded by real normal behaviour in the autoencoder.

7. Autoencoder Design and Anomaly Scoring

The autoencoder serves as the unsupervised anomaly detector[1]. Its role is to model the manifold of normal behavioural vectors so that any window lying far from that manifold produces a high reconstruction error. The model used is a dense feedforward autoencoder; choices in depth and width are intentionally conservative for initial experiments, ensuring training stability and efficient inference during streaming. The autoencoder is trained exclusively on windows labeled as normal, using normalized input features to stabilize learning.

Design choices included using a Mean Squared Error (MSE) reconstruction loss, a smaller latent size relative to input dimension to encourage compressive representation, and early stopping to prevent overfitting. The reconstruction error for each sample is computed as the mean squared difference across features between input and output. To select a practical anomaly threshold, the distribution of reconstruction errors on held-out normal data is analyzed and a percentile-based cutoff is chosen (commonly the 95th percentile). This threshold is interpretable: it designates the top X% of normal-reconstruction errors as “unusual” and sets expectations for false-positive rates under normal operation.

The autoencoder’s output is used in two ways. First, as a binary anomaly indicator (error > threshold) for operational alerting. Second, and more powerfully for the hybrid model, the continuous reconstruction error is appended as an additional feature for the Random Forest, providing a learned measure of how “normal” a window appears. Empirically, using the anomaly score as an input to the supervised classifier improved recall for classes that shared only weak discriminative signals in the engineered features alone.

High-level diagram , Autoencoder training and scoring:

```
Feature matrix X_normal (n_samples x d_features)
        |
        ▼
Train Autoencoder on X_normal:
    Encoder -> latent -> Decoder
        |
        ▼
For each sample x:
    x_hat = Autoencoder.predict(x), recon_error = mean_square_error
        |
        ▼
Save recon_error (continuous) and anomaly_flag (error > percentile_threshold)
```

Practically, the autoencoder is fast to evaluate and can be executed at inference time for each streamed behavioural window. The threshold and scaler are saved alongside the model to ensure reproducible scoring in production.

8. Random Forest Classifier and Hybrid Integration

The Random Forest classifier is the supervised backbone responsible for classifying windows into fraud classes or “normal.” The classifier is trained using the engineered behaviour features plus the continuous anomaly score from the autoencoder. The forest was chosen for several pragmatic reasons: it handles mixed-type features well, is robust against irrelevant features, can be trained quickly on tabular data, and its output can be explained via SHAP. Moreover, Random Forests can incorporate class weights to mitigate class-imbalance effects typical in fraud datasets.

Model training follows a careful evaluation protocol. Data that contained classes with fewer than two examples was filtered out prior to stratified splitting to avoid invalid folds in cross-validation. An 80/20 train/test split with stratification was used to preserve class balance in the hold-out set. Cross-validation (via KFold or StratifiedKFold depending on label distribution) was used only on the training data for hyperparameter tuning. `RandomizedSearchCV` was applied over reasonable parameter ranges (n_estimators, max_depth, class_weight) to find robust configurations while limiting computational cost.

The Random Forest’s output is a class label and associated class probabilities. These probabilities are useful for triaging: high-probability fraud cases can be prioritized for manual review, while lower-probability alerts can be batched or scored for secondary checks. Combining the autoencoder anomaly score with RF features was found to improve detection rates in experiments; the AE captures deviations that simulated fraud classes may not fully represent, while RF specializes in discriminating among known patterns.

High-level diagram , Random Forest training and inference:

```
Training:
    Input: Feature vectors (behaviour features + recon_error), labels
    Train RandomForest with CV + RandomizedSearch
    Save best_estimator

Inference:
    For each new window:
        compute features
        compute recon_error via AE
        x_augmented = features + [recon_error]
        y_pred, y_proba = RandomForest.predict(x_augmented)
```

9. Explainability: SHAP Integration

Operational trust and regulatory compliance require explanations for automated decisions. This system integrates two complementary explainability tools. SHAP (SHapley Additive exPlanations) offers theoretically grounded, consistent attribution of feature contributions for both individual predictions and global model behaviour. SHAP summary plots reveal which engineered behaviour features drive the classifier across the dataset, while SHAP dependence plots help analyze how specific features (e.g., amounts_CV or avg_time_gap) affect predicted fraud risk at different values.

Explainability outputs are stored with alerts so that each flagged window includes a persistent explanation supporting downstream investigations and potential regulatory audits.

10. Deployment, Streaming Inference, and Monitoring

The pipeline was explicitly designed for streaming and operational deployment. Two components are critical for production readiness: a memory-efficient ingestion and transformation layer, and a low-latency scoring layer for per-window inference. For ingestion, the PyArrow streaming approach and Parquet row-group organization facilitate incremental reprocessing and re-use by downstream services. DuckDB (optional) may be used for SQL-like batch queries on Parquet files, providing a lightweight ad-hoc analytics layer.

For scoring, the autoencoder and Random Forest models are exported together with preprocessing artifacts (scaler, encoders, anomaly threshold). A small inference service can be built (e.g., a REST endpoint or a streaming microservice) that accepts a batch of transactions for an agent, constructs windows, computes features, evaluates the autoencoder to obtain recon_error, augments features, and runs the Random Forest to produce predictions and probabilities. Because the feature vectors are relatively small, this pipeline can operate with minimal memory footprint and low end-to-end latency.

Monitoring involves tracking several metrics: alert rate, false positive rate (via manual labels), model confidence distributions, and reconstruction error distributions. Drift detection is essential: if the distribution of recon_error or key features shifts substantially, the system should trigger a retraining workflow. Explainability metrics (e.g., distribution of top SHAP features) are also helpful for governance, sudden changes in which features drive decisions can indicate dataset shifts or model issues.

High-level flow, Deployment & monitoring flow:

```
Live transaction stream -> Ingestion (PyArrow / streaming API)
        |
Windowing & feature extraction -> Inference service:
    AE scoring -> RF scoring -> alert + explanations
        |
Store alert + explanation -> Case management / manual review
        |
Monitoring:
    Collect metrics (alert rate, recon_error dist, SHAP summaries)
    If drift detected -> Retrain pipeline
```

This deployment model supports both batch reprocessing and near-real-time scoring and ensures that operational teams can investigate alerts with the necessary contextual explanations.

11. Evaluation, Results, and Discussion

Evaluation is performed on held-out test data after careful train/test splits and required filtering of extremely rare classes. The metrics of interest include per-class precision and recall, the overall fraud detection rate (recall for fraud classes), confusion matrices, and ROC-AUC where relevant. The autoencoder's unsupervised performance is characterized by the distribution of reconstruction errors for normal versus fraud windows; the 95th percentile threshold provides an interpretable operating point that balances sensitivity and false-positive rates.

Empirically, the autoencoder alone produced a modest detection rate (~18% on initial experiments), which is not unexpected given a dense autoencoder architecture and the subtlety of some fraud behaviours that closely resemble normal activity. Incorporating the autoencoder's continuous anomaly score as a feature in the Random Forest substantially improved recall. The Random Forest itself delivered practical operating performance on simulated fraud classes, showing that the engineered features effectively capture the behaviour signatures encoded into those simulations.

Limitations remain: simulated fraud may not encompass all real-world tactics, and the removal of tiny classes (<2 samples) reduces granularity in evaluation for extremely rare behaviours. The autoencoder's shallow architecture limits its sensitivity to long-range patterns; a sequence model (LSTM or Transformer-based autoencoder) would likely improve performance for behaviours that evolve over windows longer than 50 transactions. Finally, class imbalance continues to be a core challenge; careful use of class-weighting, focal-loss alternatives, or synthetic minority oversampling (while avoiding leakage) are valid strategies for future iterations.

The evaluation outputs guide iterative improvements: model architecture adjustments, feature engineering refinements, simulation rule expansions, and production threshold calibration.

12. Closing Remarks and Next Steps

The design presented balances domain knowledge, engineering pragmatism, and academic rigor. Behaviour windows transform the detection problem into a tractable representation for both anomaly detection and supervised learning. PyArrow streaming and Parquet storage make the pipeline scalable; the hybrid combination of autoencoder anomaly scoring and Random Forest classification yields practical gains in detection capability while remaining interpretable via SHAP.

For future work, the most impactful directions include: adopting sequence-aware autoencoders (LSTM/Transformer), enriching features with graph-based indicators (agent–customer network centrality), implementing an online learning pipeline for model updates, expanding the real fraud dataset (collaboration with operations teams for label acquisition), and hardening the deployment for real-time operation with robust monitoring and retraining triggers.

References

[1] MacBrains, S. (2023). ***Anomaly Detection Using Autoencoders: A Deep Dive into Fraud Detection*****.** **https://medium.com/@stacymacbrains/anomaly-detection-using-autoencoders-a-deep-dive-into-fraud-detection-9f59bcb5ab32**

[2] Ojo, A. (2025). ***Machine Learning Approaches for Fraud Detection: A Systematic Review*****. F1000Research, 14:664.** https://f1000research.com/articles/14-664/pdf