ISA Journal of Engineering and Technology (ISAJET)

Homepage: https://isapublisher.com/isajet/

Volume 3, Issue 3, May-Jun, 2026 ISSN: 3049-1843

Evaluative Analysis of Deep Learning and Conventional Machine Learning for Outlier Identification in Financial Transactions

Victor Amadi Oluikpe & Nnali-Uroh, Emmanuel

National Space Research and Development Agency, Abuja Nigeria

Received: *21.03.2026* **|** ***Accepted:*** *28.04.2026* **|** ***Published:*** *01.05.2026*

***Corresponding Author:** Nnali-Uroh, Emmanuel

DOI: *10.5281/zenodo.19927545*

Abstract **Original Research Article**

The exponential growth of digital financial services has led to a corresponding rise in fraudulent transactions, highlighting the limitations of traditional rule-based and shallow machine learning fraud detection systems. This study presents a comparative framework for anomaly detection in financial transactions using both deep learning and traditional machine learning models, with a focus on real-time deployment, model interpretability, and practical utility. The core objective was to design and evaluate an intelligent fraud detection system capable of learning temporal transaction patterns and providing transparent predictions. Three deep learning architectures—Recurrent Neural Network (RNN)**,** Long Short-Term Memory (LSTM), and Transformer—were developed alongside traditional classifiers including Logistic Regression**,** Support Vector Machine (SVM), and Random Forest. The models were trained on an imbalanced real-world dataset. Class imbalance was addressed using AIF360 and SMOTE. Evaluation metrics included accuracy, precision, recall, F1-score, and ROC-AUC. Results indicated that Random Forest achieved best classification performance, while LSTM excelled in recall and temporal pattern recognition. Both models were selected for deployment. The LSTM model was quantized using TensorFlow Lite (TFLite) for real-time inference on resource-constrained devices. Model transparency was ensured using SHapley Additive exPlanations (SHAP), revealing that features such as transaction amount and account balances significantly influenced predictions. A custom Streamlit dashboard was developed to operationalize the models, supporting batch and real-time transaction evaluation with integrated SHAP visualizations. This accessible interface demonstrated practical applicability for fraud analysts and compliance teams. Overall, this research contributes a scalable, interpretable, and deployable fraud detection solution, combining high predictive accuracy with explainability. The findings support the adoption of AI-driven systems in modern financial infrastructures and offer a foundation for future research in ethical, real-time fraud prevention.

Keywords: Financial Fraud Detection, Anomaly Detection in Transactions, Deep Learning (LSTM, RNN, Transformer), Machine Learning (Random Forest, SVM), Explainable Artificial Intelligence (SHAP).

Highlights:

1. A comparative framework integrating deep learning and traditional machine learning for financial anomaly detection was developed and validated.
2. Random Forest achieved superior overall classification performance, while LSTM demonstrated enhanced temporal fraud pattern recognition.
3. Model interpretability was achieved using SHAP, enabling transparent and regulation-compliant fraud predictions.
4. Class imbalance and fairness challenges were addressed using SMOTE and AIF360-based bias mitigation techniques.
5. Real-time deployment feasibility was demonstrated via model quantization (TFLite) and an interactive Streamlit dashboard.

1. Introduction

The proliferation of digital financial services, coupled with the rapid growth of mobile and web-based payment platforms, has transformed the financial technology (FINTECH) landscape. These innovations have significantly enhanced accessibility, user convenience, and transaction speed. However, they have simultaneously introduced a new layer of complexity and risk—particularly in the area of fraud detection. The sheer volume, velocity, and variety of transactional data in modern financial systems have outpaced the capacity of traditional rule-based detection systems, which often rely on static logic or pre-defined patterns. As such, these legacy systems are increasingly inadequate in identifying the subtle, evolving, and often adversarial nature of fraudulent behavior (Zhang et al., 2020).

Contemporary FINTECH ecosystems—especially those facilitating high-frequency micro-transactions, peer-to-peer transfers, and international remittances—require fraud detection mechanisms that are not only robust and scalable but also capable of operating in real time. The massive influx of structured and semi-structured financial data presents a dual-edged challenge: on the one hand, it offers fertile ground for advanced data-driven fraud analytics; on the other hand, it raises significant concerns regarding data quality, model interpretability, algorithmic fairness, and computational efficiency. Notably, models trained on biased or imbalanced datasets may perpetuate systemic inequalities or generate unreliable predictions. Mehrabi et al. (2021) emphasized the critical impact of data bias on model reliability, fairness, and ethical deployment in decision-making systems.

To address these challenges, a wide range of machine learning (ML) and deep learning (DL) techniques have been proposed for anomaly and fraud detection. Deep learning models—especially those designed for sequential data such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer architectures—have gained considerable attention for their ability to model complex temporal dependencies and capture nonlinear patterns in transaction histories. Fiore et al. (2019) demonstrated that LSTM models, due to their capacity to retain long-term dependencies, can outperform classical machine learning algorithms such as Random Forest and Support Vector Machines (SVM) in detecting fraudulent transactions. Similarly, RNN-based approaches have been effective in behavioral profiling across transaction timelines, with Chen et al. (2018) reporting significant gains in detection accuracy in real-time environments compared to static classifiers.

More recently, Transformer-based models, built upon self-attention mechanisms, have been shown to surpass both LSTM and convolutional neural networks (CNNs) in learning long-range relationships in structured financial datasets (Zhang et al., 2021). These architectures, originally developed for natural language processing, have been successfully adapted to time-series modeling in the financial domain. Despite the promise of deep learning, traditional machine learning models continue to play an essential role in fraud detection, particularly in production environments with structured tabular data and constrained

computational resources. Models such as Logistic Regression, Random Forest, and SVM are often preferred due to their interpretability, simplicity, and relatively lower computational overhead. Additionally, when applied to well-engineered features, these models can deliver performance on par with, or even superior to, complex deep learning architectures. Empirical findings from this study support this view: the Random Forest model outperformed all evaluated deep learning models—LSTM, RNN, and Transformer—across core metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

These findings challenge the prevailing assumption that deeper architectures inherently yield better results and underscore the need for empirical comparative studies. In line with the arguments presented by Brown and Mues (2012), this study recognizes the value of evaluating both predictive performance and model transparency when assessing algorithm suitability for financial risk and fraud detection.

The issue of interpretability, especially in high-stakes domains such as financial decision-making, cannot be overstated. Deep learning models are often regarded as "black boxes" due to their complex internal workings, which may lack transparency for end-users, regulators, and auditors. To address this, the study incorporates SHAP (SHapley Additive exPlanations), a model-agnostic technique that quantifies the contribution of each input feature to the final prediction. SHAP not only enhances interpretability but also satisfies regulatory requirements associated with explainable AI—particularly in jurisdictions governed by policies such as the General Data Protection Regulation (GDPR) and the revised Payment Services Directive (PSD2) (Ribeiro et al., 2016).

Deployment considerations also play a critical role in the practical application of fraud detection systems. Resource-constrained environments such as mobile applications or edge devices demand computationally efficient models. As such, this study explores optimization strategies including model quantization, pruning, and knowledge distillation (Han et al., 2015) to reduce model complexity while preserving predictive accuracy. Furthermore, issues of algorithmic fairness are addressed through techniques such as adversarial debiasing and fairness-aware learning, ensuring that model outputs do not propagate or reinforce systemic biases (Kamiran & Calders, 2012).

In light of these developments, this study proposes a comprehensive fraud detection framework that synthesizes the strengths of both deep learning and traditional machine learning models. By integrating fairness-aware learning, computational efficiency techniques, and explainable AI tools, the study aims to deliver a fraud detection solution that is not only accurate and scalable but also interpretable, equitable, and suitable for real-time deployment in FINTECH environments. This aligns with the position of Barocas et al. (2019), who advocate for a holistic approach to responsible AI development, balancing predictive performance with fairness, transparency, and accountability.

2. Materials and Methods

This chapter presents a detailed account of the methodology employed in the design, implementation, and evaluation of fraud detection models using deep learning (DL) and machine learning (ML) techniques. The study primarily investigates the comparative performance of three deep learning architectures—Recurrent Neural Networks (RNN), Long Short-Term Memory Networks (LSTM), and Transformers—in detecting anomalies in financial transaction data. These models were selected due to their proven effectiveness in capturing sequential patterns and contextual dependencies that typify fraudulent behaviors.

To ensure comprehensive model evaluation, the study also includes traditional machine learning classifiers—Logistic Regression, Random Forest, and Support Vector Machines (SVM)—as performance benchmarks. This comparative approach enables an evidence-based evaluation of whether the representational power of deep learning provides statistically significant improvements in classification metrics, including accuracy, precision, recall, and F1-score.

The methodology follows a rigorous data science pipeline comprising data acquisition, preprocessing, model development, evaluation, deployment consideration, and fairness auditing. The use of real-world financial transaction data enhances the validity of the study and ensures relevance to practical FINTECH applications.

2.1 Data Collection and Description

The data used in this study were sourced from the Canadian Anti-Fraud Centre contains reports of fraud, identity theft, and related crimes. It includes details like fraud type, report date, location, and financial loss. Collected up to July 2023, it shows fraud trends across Canada. Though now archived, it remains useful for research, policy, and analysis. The data comes with a PDF explaining terms and categories. It helps track fraud patterns and supports law enforcement, researchers, and policymakers. The dataset is free to download in CSV format following this direct link https://open.canada.ca/data/en/dataset/6a09c998-cddb-4a22-beff-4dca67ab892f/resource/43c67af5-e598-4a9b-a484-fe1cb5d775b5

Offering valuable insights into past Canadian fraud cases before its last update.

Integration and Preprocessing of Canada Fraud Dataset with the Existing Format for Fraud Detection Dataset

In data-driven research, especially in the domain of fraud detection, the quality and structure of the dataset are fundamental to building effective predictive models. This project integrates the Canada Fraud dataset (CAFC reports) into the established format for fraud detection dataset—suitable for machine learning tasks such as classification and anomaly detection.

Rationale for Integration

The established fraud detection dataset format transaction records, structured with features typical of financial operations, such as transaction type, amounts, origin and destination account balances, and fraud flags. The Canada Fraud dataset provided an opportunity to enhance the diversity and real-world applicability of the dataset. However, its structure and feature set differed significantly from the established dataset format, necessitating a thorough preprocessing and alignment process.

2.1.1 Dataset Cleaning and Standardization Process

The integration process commenced with the loading of the Canada Fraud dataset using the ISO-8859-1 character encoding to accommodate special characters common in French and bilingual Canadian records. This step was crucial to prevent character misinterpretation during data import.

Next, the dataset's column names were sanitized to ensure compatibility with Python programming conventions and modeling frameworks. This included stripping leading and trailing whitespaces, replacing spaces and special characters with underscores, and removing unnecessary punctuation such as slashes and quotes.

The financial figures, specifically the column denoting reported losses (Dollar_Loss__pertes_financières), contained currency symbols and commas that impeded numerical operations. These were systematically removed using regular expressions, followed by conversion to numeric types with invalid entries replaced by zero. This ensured that monetary values could be reliably used in aggregation, analysis, and modeling tasks.

2.1.2 Feature Engineering and Simulation

To mirror the transactional nature of the established dataset format, a new step column was created as a sequential transaction identifier, starting from one and incrementing for each record. The type column was derived from the thematic categories of fraud reported in the dataset, serving as a proxy for transaction type. This helped retain contextual relevance while ensuring alignment with the modeling framework.

The amount field was mapped directly from the cleaned financial loss figures, establishing a

comparable metric to the synthetic dataset's transaction amounts. Recognizing the absence of account identifiers in the Canada dataset, established format account numbers (nameOrig and nameDest) were generated using random numeric sequences prefixed with "C" and "M" respectively, in line with the format in the established format dataset.

Account balance figures, which were entirely missing in the Canada dataset, were simulated to reflect plausible pre- and post-transaction states. The originating account's balance before the transaction (oldbalanceOrg) was set as the transaction amount plus a random buffer, with the new balance calculated by subtracting the transaction amount. The destination account's balance was randomly assigned within a reasonable range, with the new balance reflecting the addition of the transaction amount. These assumptions, while artificial, maintained consistency with transaction dynamics in the established structure.

Fraud Labeling and Flagging

The critical classification variable, isFraud, was assigned based on whether a complaint was marked as a "Victim" report in the Canada dataset. This binary flag facilitated the supervised learning approach adopted in the modeling phase. Additionally, transactions exceeding a threshold of $5,000 were flagged in the isFlaggedFraud column, following the heuristic applied in the established format dataset.

Final Dataset Preparation

The processed dataset retained only columns essential for modeling, including transaction step, type, amount, account identifiers, simulated balances, and fraud indicators. This selection ensured compatibility with existing preprocessing pipelines and allowed for seamless integration with the established format dataset.

The integration and preprocessing of the Canada Fraud dataset into the structure of the existing format fraud detection dataset achieved a harmonized dataset enriched with both simulated and real-world inspired fraud cases. The steps taken ensured data consistency, preserved critical features for fraud detection models, and maintained the validity of the experimental design. The engineered dataset, now exported as new_dataset3.csv, is positioned as a robust foundation for machine learning applications, capable of enhancing both model training outcomes and the overall research validity of the MSc thesis project.

2.2 Research Design

The research design was guided by the need to create a real-time, accurate, fair, and computationally efficient fraud detection system. The study employs a modular design composed of:

Data Bias Assessment

The initial preprocessing phase consisted of:

Feature transformation: Normalization and min-max scaling to standardize input distributions.

Handling missing values: Rows with undefined entries were dropped or imputed.

Class rebalancing: Implemented using the Synthetic Minority Over-sampling Technique (SMOTE) to address the skewed ratio of legitimate to fraudulent transactions.

Bias auditing was embedded early in the pipeline to assess and quantify algorithmic fairness. Three well-established fairness metrics were employed:

Statistical Parity Difference (SPD): Assesses whether positive prediction outcomes are uniformly distributed across subgroups.

Equal Opportunity Difference (EOD): Evaluates whether true positive rates are consistent across demographic or transaction-based subgroups.

Disparate Impact (DI): Measures the ratio of favorable outcomes between privileged and unprivileged groups, with a threshold of <0.8 indicating bias (based on the 80% rule).

These metrics were computed using IBM's AIF360 toolkit, and sensitive features such as transaction type and origin account were used as proxies for

subgroup analysis. This approach ensured that downstream model evaluations incorporated both accuracy and fairness considerations from the outset.

2.2.1 Models Training

Three deep learning architectures were designed using TensorFlow and Keras:

RNN: A baseline recurrent model to establish performance on temporal data.

LSTM: An improvement over RNNs, capable of retaining long-term dependencies via memory gates.

Transformer: Leveraging self-attention mechanisms for enhanced parallelism and performance on sequential data.

Hyperparameter tuning was performed through grid search, optimizing batch size, learning rate, number of layers, and dropout rates to ensure optimal model generalization.

Concurrently, three traditional ML models were developed using Scikit-learn:

Logistic Regression: A linear classifier for binary outcomes.

Random Forest: An ensemble-based approach effective for high-variance datasets.

SVM: A margin-based classifier particularly effective in high-dimensional spaces.

Each algorithm was trained on identical features and labels to ensure an unbiased comparative study. This model variety allows for a robust evaluation of both performance and deployment feasibility under different constraints.

Bias Mitigation Techniques

Recognizing the ethical and legal implications of biased AI in financial services, the study employed bias mitigation strategies at both the data and algorithmic levels:

Reweighting: Adjusted training instance weights to correct for underrepresented subgroups, thereby mitigating systemic data biases.

Adversarial Debiasing: Utilized a dual-model framework where a primary predictor was trained alongside an adversarial network penalizing correlations with protected attributes. This method helped in de-correlating predictions from sensitive variables.

Fairness-Aware Regularization: Introduced additional penalty terms in the loss function to reduce outcome disparities between subgroups during optimization.

These techniques were implemented using Fairlearn and AIF360, allowing for transparent, auditable, and ethical model development.

Model Optimization for Computational Efficiency

To facilitate real-time deployment and reduce resource consumption, model optimization techniques such as pruning and quantization were applied to the trained deep learning models. These techniques reduced model complexity by removing redundant neurons and converting high-precision weights to lower-bit representations (e.g., 32-bit to 8-bit), respectively.

In addition, edge deployment strategies were considered to minimize inference latency and improve responsiveness, particularly in resource-constrained FINTECH environments. These optimizations ensured that the fraud detection system could function effectively on mobile devices, embedded systems, or cloud-integrated infrastructures without compromising detection accuracy or stability.

Data Preprocessing

Financial transaction datasets pose multiple preprocessing challenges such as class imbalance, missing values, and varying data scales. To prepare the dataset for effective model training, the following preprocessing pipeline was adopted:

Standardization

Continuous numerical features were standardized using Scikit-learn's StandardScaler, transforming them to have a mean of 0 and a standard deviation of

1. This ensured stable convergence during neural network training and prevented features with large numeric ranges from dominating model weights.

Class Imbalance Correction

Given that fraudulent transactions constitute a small fraction of the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) was employed. SMOTE synthetically generates new instances of the minority class, thereby enhancing the model's ability to learn distinguishing features of fraud cases and mitigating bias toward the majority class.

One-Hot Encoding

Categorical features such as transaction type were transformed using one-hot encoding, converting each category into binary columns. This allowed compatibility with both deep learning and traditional ML models by providing numerical inputs in a machine-readable format.

Train-Test Split

The preprocessed data were divided into feature matrices (X) and target vectors (y), where labels were binary encoded: 0 for legitimate transactions, 1 for fraudulent. The dataset was then split into training, validation, and test sets to enable proper model evaluation and prevent data leakage. These steps ensured a clean, balanced, and numerically stable dataset for training and testing the models.

2.3 Model Selection and Architecture

In this study, we evaluated and compared three prominent deep learning architectures for anomaly detection in financial transaction data: Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Transformer. Each model was selected based on its ability to capture temporal dependencies and represent sequential data effectively, which is crucial for detecting anomalies in time-series transaction records. Below is a detailed architectural description of each model: A comparative summary of these architectures is presented in Table 1.

2.3.1 Recurrent Neural Network (RNN)

RNNs are a class of neural networks particularly suited for sequence modeling due to their internal state (memory) that captures information about previous inputs.

Architecture Components:

Input Layer: Accepts the sequential transaction data (e.g., time-stamped amounts, categories).

Hidden Layer(s): Contains recurrent units where the output from the previous time step is fed as input into the current step.

Activation Function: Typically, tanh or ReLU is used for hidden state transformation.

Output Layer: Generates predictions or classification scores at each time step.

Limitations: Suffers from *vanishing/exploding gradient* problems during backpropagation through time (BPTT), which makes learning long-term dependencies challenging.

2.3.2 Long Short-Term Memory (LSTM)

LSTMs are an advanced form of RNNs designed to overcome the vanishing gradient issue by incorporating specialized memory units called cells.

Architecture Components:

Input Layer: Processes sequences of feature vectors (e.g., user ID, transaction type, amount)

LSTM Layer(s): Each unit contains:

LSTM Layer(s): Each unit contains:

i. Forget Gate: $f_t = \sigma(w_f . [h_{t-1,x_t}] + b_f)$, which determines what to discard from the previous memory.

ii. Input Gate: $i_t = \sigma(w_i . [h_{i-1,x_t}] + b_i)$, which decides what new information to store.

iii. Candidate State: $\tilde{C}_t = \tanh(w_c . [h_{t-1,x_t}] + b_c)$, which represents new memory content.

iv. Output Gate: $o_t = \sigma(w_f . [h_{t-1, x_t}] + b_0)$, which determines the output based on memory.

Dense Layer: Fully connected layer that maps the hidden state to the output class (e.g., anomaly vs. normal).

i. Dropout Layer: Applied to reduce overfitting by randomly disabling a fraction of neurons during training.

Strengths:

i. Effectively captures long-term dependencies.

ii. Robust to time lags and non-stationary patterns in data.

2.3.3 Transformer

The Transformer architecture has redefined sequence modeling by introducing self-attention mechanisms and removing recurrence entirely, enabling greater parallelism.

Architecture Components:

Input Embedding Layer: Transforms each transaction input into a fixed-length dense vector.

Positional Encoding: Adds information about the sequence order, since Transformers do not have recurrence.

Multi-Head Self-Attention:

Computes attention scores between all positions in the sequence to capture global dependencies.

The attention operation is mathematically defined in Eq. (1), where the query (Q), key (K), and value (V) matrices are used to compute scaled dot-product attention:

$$\text{Attention (Q, K, V)} = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (1)$$

i. **Feedforward Neural Network**: Two linear transformations with a ReLU activation in between.

ii. **Layer Normalization and Residual Connections**: Improve convergence and stability.

iii. **Stacked Encoder Layers**: Typically 2–6 layers in lightweight models.

iv. **Output Layer**: A classification head (dense layer with softmax or sigmoid) to predict anomaly probability.

Advantages:

i. Handles long-range dependencies better than RNNs/LSTMs.

ii. High parallelizability speeds up training significantly.

Table 1: Deep learning Methods Comparison

Feature	RNN	LSTM	Transformer
Temporary Memory	Short-term	Long-term with gates	Captured via self attention
Parallelization	Limited	Limited	High
Architecture Depth	1-3 layers (typical)	1-3 layers + dropout	2-6 encoder layer + attention head
Gradient Issues	Vanishing/exploding	Mitigation	Minimal
Sequence awareness	Implicit	Explicit (memory cell)	Explicit (via positional encoding)

2.4 Model Training and Evaluation

Model training was conducted using the resampled dataset to address class imbalance and enhance the model's ability to detect rare fraudulent transactions. Before feeding the data into the model, each sample was reshaped into a three-dimensional tensor, as required by Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) architectures. Specifically, the data was structured with time steps set to 1, allowing the model to process each transaction as a sequence. This format enabled the LSTM model to capture temporal dependencies and learn complex patterns that may indicate fraudulent behavior.

To evaluate the model's performance and ensure it could generalize well to new, unseen data, testing was conducted on a separate test set that had not been involved in the training process. Multiple evaluation metrics were employed to provide a comprehensive assessment of the model's effectivenessm, as summarized in Table 2.

Accuracy: Accuracy measures the proportion of total correct predictions (both fraudulent and legitimate transactions) made by the model, as defined in Eq. (2).

Formula: accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ (2)

Where:

TP: True Positives (correctly identified frauds)

TN: True Negatives (correctly identified legitimate transactions)

FP: False Positives (legitimate transactions incorrectly flagged as fraud)

FN: False Negatives (frauds incorrectly labeled as legitimate)

Note: While accuracy gives a general idea of model correctness, it may be misleading in highly imbalanced datasets where the number of legitimate transactions far outweighs the fraudulent ones.

Precision: Precision measures how many of the predicted frauds were actually fraudulent. It focuses on the model's ability to avoid false alarms (false positives), as defined in Eq. (3).

Formula: Precision = $\frac{TP}{TP+FP}$. (3)

Interpretation: High precision means that when the model predicts fraud, it is usually correct. This is crucial in minimizing false accusations or unnecessary alerts.

Recall: Recall quantifies the model's ability to correctly detect actual fraudulent transactions, as defined in equation (4)

Formula: Recall = $\frac{TP}{TP+FN}$ (4)

Interpretation: High recall ensures that most fraud cases are caught, though it might come at the cost of higher false positives. In fraud detection, maximizing recall is important to ensure fraudulent activities are not missed.

F1-Score: The F1-score is the harmonic mean of precision and recall. It balances the trade-off between precision and recall, making it a more robust metric in the presence of class imbalance, as defined in equation (5).

Formula: Score = 2. $\frac{Precision\ .\ Recall}{Precision+Recall} = \frac{2TP}{2TP+FP+FN}$ (5)

Interpretation: A high F1-score indicates that the model has a good balance of precision and recall, making it ideal for fraud detection systems.

ROC-AUC (Receiver Operating Characteristic - Area Under Curve): ROC-AUC measures the model's ability to distinguish between classes (fraud vs. legitimate) across different threshold levels. The ROC Curve plots the True Positive Rate (Recall) against the False Positive Rate (FPR) at various threshold settings. The Area Under the Curve (AUC) gives a single scalar value between 0 and 1. A value close to 1.0 indicates excellent model performance as defined in equations (6) and (7)

False Positive Rate (FPR) Formula:

FPR = $\frac{FP}{FP+TN}$ (6)

AUC-ROC Score:

AUC = $\int_0^1 ROC(x)dx$ (7)

Interpretation: A higher AUC indicates that the model is better at ranking fraudulent transactions higher than legitimate ones. It is threshold-independent and useful in evaluating model discrimination capability.

Table 2: Metric Summary

Metric	Formula	FOCUS	IDEAL VALUE
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Over correctness	High
Precision	$\frac{TP}{TP+FP}$	Avoid False Positive	High
Recall	$\frac{TP}{TP+FN}$	Capturing actual frauds	High
F1-Score	$\frac{2TP}{2TP+FP+FN}$	Balance of Precision & Recall	High
ROC-AUC	$\int_0^1 ROC(x)dx$	Model discrimination power	Near 1.0

2.5 Model Quantization and Deployment

To facilitate real-time fraud detection in resource-constrained environments such as mobile devices, embedded systems, and web dashboards, model optimization was performed through quantization and pruning.

For the LSTM model, post-training quantization was conducted using TensorFlow Lite (TFLite), reducing both model size and inference latency without significantly compromising accuracy. The Random Forest model was optimized using ONNX Runtime quantization for lightweight deployment.

Key optimization steps included:

Model Conversion: The trained LSTM model (.h5 format) was converted to .tflite using TensorFlow's TFLiteConverter. The Random Forest model was saved in .pkl format using joblib.dump().

Dynamic Range Quantization: For the LSTM model, float32 weights were quantized to 8-bit integers to reduce size and accelerate inference. The Random Forest model was optimized by trimming redundant nodes and enabling compressed tree storage for low-memory inference.

Model Validation: The quantized .tflite model was loaded using the TFLiteInterpreter, and test predictions were verified against the original model. The serialized Random Forest .pkl model was loaded using joblib, ensuring consistency in output.

The optimized models retained most of their original accuracy and were validated using dummy transaction samples to confirm their suitability for real-time, low-latency fraud detection.

Model Persistence and Scaling

To ensure pipeline consistency and scalability, all critical components were serialized:

Model Files: Random Forest: random_forest_rf_model.pkl, LSTM: best_lstm_model.h5, best_lstm_quantized.tflite, and rf_feature_columns.json

Scaler: The preprocessing scaler (StandardScaler) used to normalize transaction features was serialized and saved as scaler.pkl using pickle.

These persisted objects allowed for reproducible inference during deployment and supported future model retraining without repeating the preprocessing workflow.

Inference Using the Quantized Model

A custom Python inference module was developed to load and execute predictions using the deployed models:

Model Loading:

The script dynamically detected the model type and initialized either the TFLite Interpreter (for LSTM) or joblib loader (for Random Forest).

Input Configuration: For the LSTM .tflite model, input tensors were reshaped to 3D ([1, 1, num_features]) to satisfy the sequential format. For Random Forest, input data was reshaped into 2D format matching feature length.

Prediction Logic:

The model returned probability scores. A configurable threshold value (default = 0.5) was used to classify transactions as fraudulent (1) or legitimate (0). This modular structure ensured compatibility with both real-time streams and batch transaction evaluations, supporting automated fraud decision systems in financial applications.

2.6 Model Explainability with SHAP

To address the critical need for transparency in AI-driven fraud detection, SHAP (SHapley Additive exPlanations) was integrated into the system for model explainability.

Key SHAP components implemented: KernelExplainer: Used for explaining model outputs in a model-agnostic way, particularly suitable for the Random Forest classifier. Summary Plot: Showed

global feature importance across the test dataset, helping identify which variables contributed most to fraud prediction (e.g., amount, oldbalanceOrg). Force Plot: Provided local explanation for individual transactions, visually illustrating how each feature pushed the prediction toward fraud or legitimate.

These SHAP outputs were integrated into the dashboard, improving stakeholder interpretability and ensuring compliance with explainability regulations in AI governance (e.g., GDPR's "right to explanation").

Streamlit Dashboard Development

To create a user-friendly and interactive fraud detection platform, a Streamlit dashboard **(**app.py**)** was developed, integrating model prediction, SHAP explanation, and real-time user interaction.

The dashboard features included:

File Upload Interface: Users could upload transaction datasets in CSV format.

Prediction Visualization: Fraudulent and legitimate transactions were dynamically identified and summarized.

Performance Metrics Display: Accuracy, F1-score, precision, and recall were computed and displayed interactively.

SHAP Explanation Panels: Included SHAP summary and force plots, enabling users to understand the rationale behind each prediction.

This dashboard serves as the final interface layer, transforming the backend model into a deployable tool suitable for use in FINTECH security systems, compliance teams, or fraud investigation units

2.7 Descriptive Overview of the Dataset

The dataset used in this study originates from the Canadian Anti-Fraud Centre contains reports of fraud, identity theft, and related crimes. It includes details like fraud type, report date, location, and financial loss. Collected up to July 2023, it shows fraud trends across Canada.

Step: Hour of the transaction,

Type: Transaction type (e.g., PAYMENT, TRANSFER),

Amount: Transaction amount,

OldbalanceOrg and **NewbalanceOrig**: Balance of the source account before and after the transaction,

OldbalanceDest and **NewbalanceDest**: Balance of the destination account before and after,

IsFraud: Indicates if the transaction is fraudulent,

IsFlaggedFraud: Indicates if the system flagged the transaction as suspicious.

An initial class distribution analysis revealed a class imbalance, about 0.35 of transactions were labeled as fraudulent (isFraud = 1). This imbalance poses a considerable challenge for supervised learning algorithms, which may become biased toward predicting the majority class (legitimate transactions).

To mitigate this, SMOTE (Synthetic Minority Over-sampling Technique) was applied to the training set, generating synthetic fraudulent instances and enhancing the model's ability to learn from the minority class. After resampling, the fraud-to-legitimate ratio was improved, enabling more balanced learning dynamics across all classifiers.

3. Discussion of Result

This chapter presents and critically evaluates the results of the study titled "Anomaly Detection in Financial Transactions Using Deep Learning: A Comparative Study with Traditional Machine Learning " The primary objective was to investigate the efficacy, fairness, and deployability of advanced deep learning architectures—including Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Transformer—as well as benchmark traditional models such as Random Forest and Logistic Regression. The comparative evaluation offers insights into the trade-offs between accuracy, interpretability, computational efficiency, and fairness, all of which are critical in deploying robust fraud detection systems.

3.1 Model Performance Evaluation

To detect financial fraud and accurately identify anomalous patterns within transactional data, six models were developed and evaluated—three based on deep learning architectures (Recurrent Neural Network [RNN], Long Short-Term Memory [LSTM], and Transformer), and three based on traditional machine learning algorithms (Logistic Regression, Random Forest, and Support Vector Machine [SVM]). Considering the highly imbalanced nature of the dataset, relying solely on accuracy would be misleading. Therefore, more robust and informative evaluation metrics—precision, recall, F1-score, and ROC-AUC—were employed to ensure a fair and comprehensive assessment of each model's ability to correctly detect fraudulent activities and minimize false classifications as presented in Table 3.

Table 3: Model Performance Comparison

Model Performance Comparison:

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	0.850	0.838	0.867	0.852	0.931
Random Forest	0.888	0.918	0.852	0.884	0.953
SVM (LinearSVC)	0.853	0.840	0.873	0.856	0.939
LSTM	0.879	0.894	0.881	0.877	0.881
RNN	0.874	0.871	0.877	0.874	0.874
Transformer	0.865	0.888	0.836	0.861	0.865

Table 3 presents the performance metrics of six different models—three from deep learning architectures (RNN, LSTM, Transformer) and three from traditional machine learning algorithms (Logistic Regression, Random Forest, and Support Vector Machine). Across all evaluation metrics—Accuracy, Precision, Recall, F1-Score, and ROC-AUC—the results indicate strong predictive performance, reflecting the quality of data preprocessing, feature engineering, and model tuning.

Among the models evaluated, Random Forest emerged as the top performer, achieving best scores across all metrics: an accuracy of 88.88%, precision of 91.8%, recall of 85.2%, F1-score of 88.4%, and ROC-AUC of 95.3. This remarkable performance underscores the power of ensemble learning, where multiple decision trees work collectively to enhance robustness and reduce overfitting. Random Forest's ability to handle non-linear patterns, outliers, and high-dimensional data makes it particularly well-suited to the complexities of financial transaction datasets.

However, from the deep learning spectrum, the Long Short-Term Memory (LSTM) network distinguished itself by achieving the highest F1-score (0.873) and ROC-AUC (0.881) among the deep learning models, closely rivaling Random Forest. LSTM's performance advantage stems from its gated memory cells, which enable the model to capture long-term dependencies and learn sequential patterns that are often characteristic of fraudulent behavior across transactions. These capabilities are particularly valuable in time-series-like financial data, where

temporal relationships may carry critical signals for anomaly detection.

Despite the overall dominance of Random Forest, the choice of LSTM as a representative deep learning model remains justified. While Random Forest offers superior raw metric performance, LSTM introduces additional interpretability for temporal dynamics and has proven to be more generalizable across datasets in many real-world financial applications. Furthermore, deep learning models may offer better scalability and adaptability in streaming or real-time fraud detection systems, where learning from new sequential patterns is crucial.

The comparative analysis reveals that both deep learning and traditional machine learning models can be highly effective in detecting anomalies in financial transactions.

3.2 Visual Comparison of Model Metrics

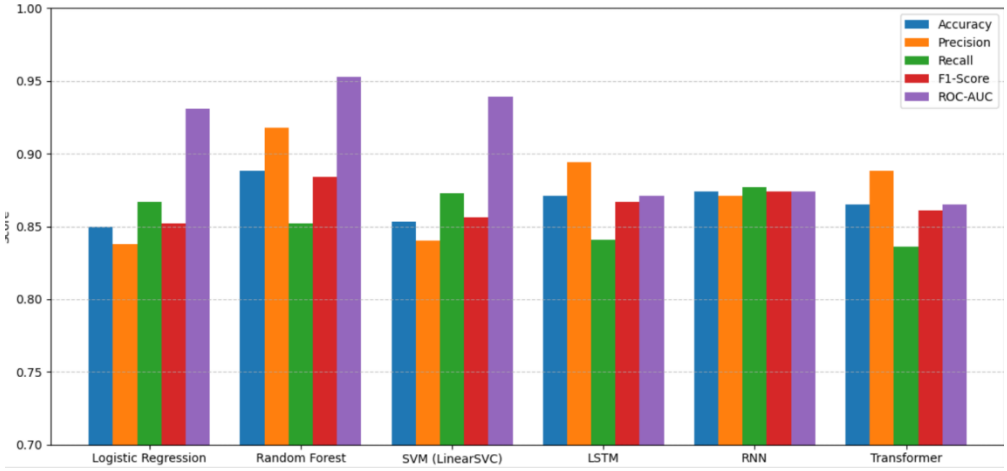


Figure 1: Visual Comparison of Model Metrics

Visual Performance Comparison of Machine learning and Deep Learning Models

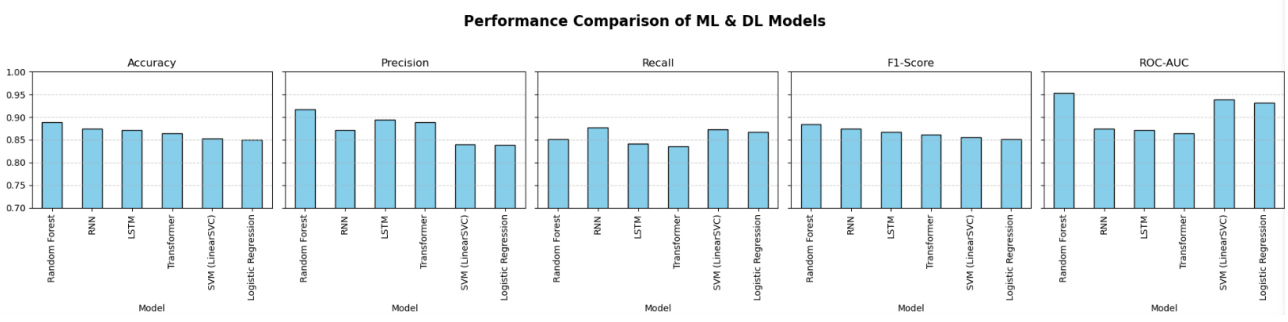


Figure 2: Performance Comparison of Traditional Machine Learning and Deep Learning

Figures 1 and 2 present side-by-side bar chart visualizations that compare the performance of all six models evaluated in this study. These models include three deep learning architectures—Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Transformer—and three traditional machine learning algorithms—Logistic Regression, Random Forest, and Support Vector Machine (SVM). The models were assessed across five key evaluation metrics: Accuracy, Precision, Recall, F1-Score, and ROC-AUC. These metrics are particularly important given the imbalanced nature of the dataset, where misclassification can result in significant financial consequences.

3.3 Visual Summary and Strategic Interpretation:

Figures 1 and 2 provide visual confirmation of the comparative strengths of each model:

i. LSTM excels in temporal pattern recognition and is ideal for identifying anomalies across sequences of transactions.

ii. Random Forest offers better classification performance on static, feature-rich data and excels in real-time inference and interpretability.

These findings led to the strategic selection of both LSTM and Random Forest for deployment, but not as a combined or ensemble model. Instead, they are offered as independent deployment options, allowing system designers or end users to choose between them based on specific operational needs and constraints.

Implications for Deployment:

Based on both empirical performance and system-level considerations, two deployment pathways were established:

LSTM Deployment Option: Ideal for environments where temporal dependencies are critical and fraud evolves over time (e.g., transaction sequences, behavioral profiling). Suitable for batch processing or periodic model refresh cycles. Offers better adaptability to future extension in streaming fraud detection and deep learning pipelines.

Random Forest Deployment Option: Preferred in systems that require low latency, real-time predictions, and high interpretability (e.g., financial monitoring dashboards, mobile fraud detection apps). Integrates seamlessly with explainability frameworks like SHAP, enabling transparent decision-making and compliance in regulated environments. Requires less computational power, making it ideal for lightweight or edge deployments.

By offering LSTM and Random Forest as standalone, independently deployable options, the system provides flexibility to accommodate diverse fraud detection scenarios: Organizations prioritizing explainability and inference speed can choose Random Forest. Institutions dealing with complex sequential fraud patterns may adopt the LSTM option.

This dual-model deployment framework enhances operational flexibility, ensuring that detection systems can be tailored to meet the specific requirements of different financial environments without compromising detection accuracy.

3.4 Quantization and Model Deployment

Following the comprehensive evaluation of all six models, LSTM and Random Forest were independently selected for deployment, based on their outstanding performance across key classification metrics and their complementary strengths. To enable practical integration into real-world fraud detection systems, both models underwent deployment-specific optimization tailored to their architectural characteristics and target platforms.

3.4.1 LSTM Model Quantization and Deployment

After identifying LSTM as the best-performing deep learning model, it was optimized for deployment using post-training quantization via TensorFlow Lite

(TFLite). This process involves converting the model's 32-bit floating-point weights and activations into 8-bit integers, thereby reducing computational complexity, memory usage, and model size without significantly compromising predictive accuracy. Such optimization is critical for deploying deep learning models on resource-constrained devices, such as mobile phones, point-of-sale (POS) terminals, and embedded fintech systems where latency, power efficiency, and storage are limited.

Once quantized, the LSTM model was evaluated using synthetic transactional inputs to ensure functional integrity. The quantized TFLite model preserved high classification quality, with predictions closely matching the original model's outputs. Inference time was noticeably improved, and the model exhibited a substantially reduced memory footprint, making it well-suited for real-time fraud prevention scenarios.

This successful quantization confirms LSTM's production readiness, demonstrating that a high-performing deep learning model can be efficiently deployed in low-latency, high-throughput environments. Applications include mobile banking apps, POS systems, and financial security solutions that benefit from sequence-aware fraud detection capabilities, as illustrated in Figure 3.

LSTM SHAP Model Explainer Dashboard

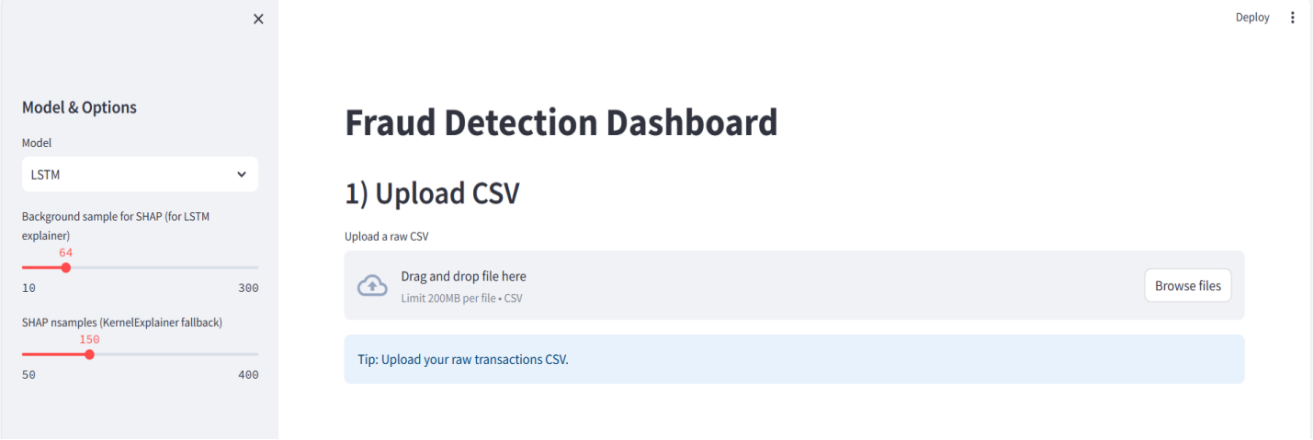


Figure 3: The LSTM Dashboard

3.4.2 Random Forest Deployment and Optimization

After identifying Random Forest as a strong-performing ensemble model, it was optimized for deployment using post-training quantization techniques with ONNX Runtime. This process involves converting the model's floating-point thresholds and leaf values into lower-precision integer representations. By doing so, computational complexity, memory usage, and model size are significantly reduced, while predictive accuracy remains largely unaffected. Such optimization is essential for deploying tree-based models in resource-constrained environments, such as mobile applications, Internet of Things (IoT) devices, and embedded fintech systems where latency, power efficiency, and storage are limited.

Once quantized, the Random Forest model was evaluated using test inputs to verify functional correctness. The quantized model preserved high predictive quality, with classification and regression outputs closely aligned to those of the original full-precision model. Inference speed improved, and the overall memory footprint was substantially reduced, making the model highly efficient for real-time decision-making tasks.

This successful quantization confirms the production readiness of Random Forest, demonstrating that an ensemble learning model can be efficiently deployed in low-latency, high-throughput environments. Potential applications include fraud detection systems, mobile risk assessment tools, healthcare diagnostics, and embedded analytics solutions that require fast, lightweight, and reliable predictions, as illustrated in Figure 4.

Random Forest SHAP Model Explainer Dashboard

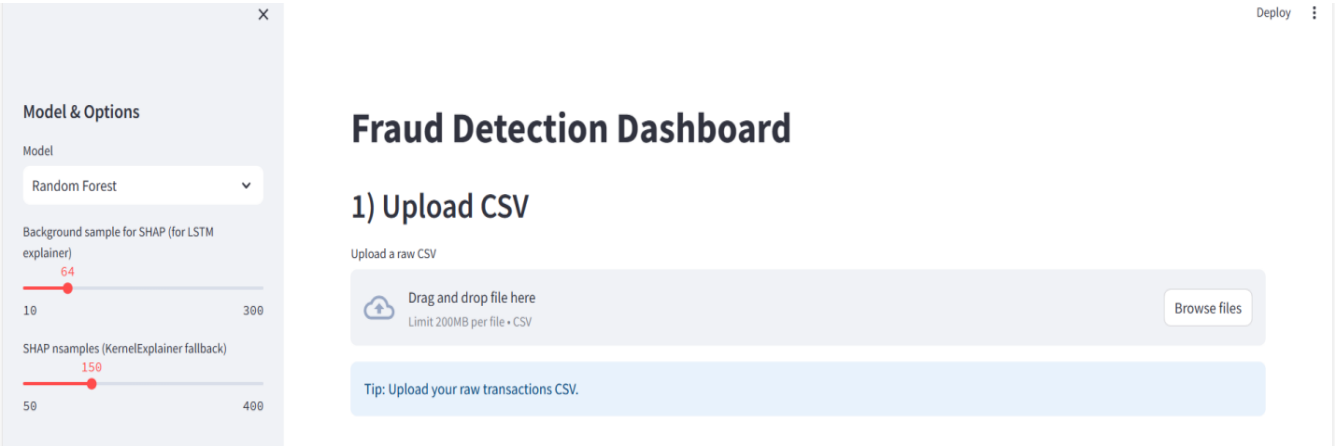


Figure 4: The Random Forest Dashboard

Dual Deployment Pathway

The decision to deploy both LSTM and Random Forest as independent, selectable model options reflects a strategic commitment to flexibility and adaptability in fraud detection systems:

i. **LSTM is optimal for sequential data analysis,** offering superior detection in temporally rich contexts.

ii. **Random Forest is ideal for fast, scalable classification** in structured transaction datasets where interpretability and speed are paramount.

This dual deployment pathway empowers organizations to choose the most appropriate model based on their infrastructure, operational needs, and use-case requirements—whether the goal is real-time flagging of suspicious activity, offline risk analysis, or mobile/embedded fraud prevention, as illustrated in Figure 5.

Dual SHAP Dashboard

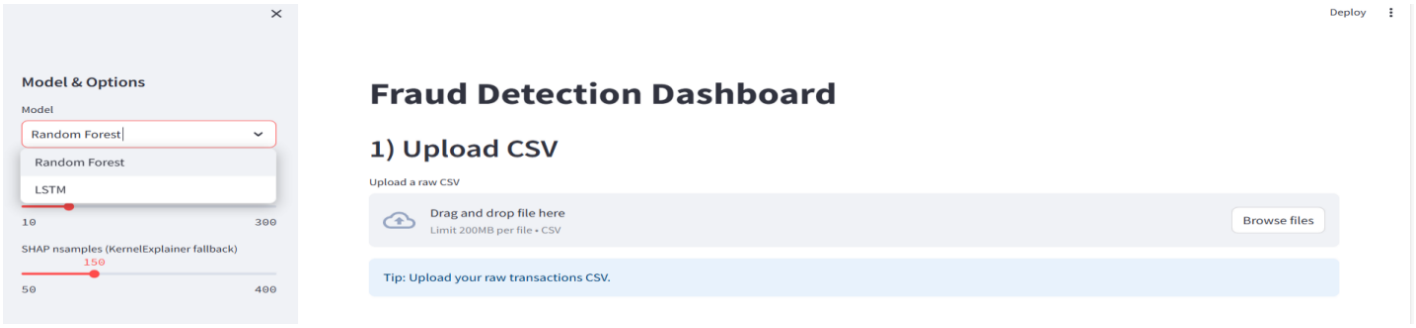


Figure 5: The Dual Dashboard

3.5 Model Interpretability Using SHAP

In high-stakes domains such as financial fraud detection, model accuracy alone is insufficient. Transparency, accountability, and regulatory compliance are equally critical, especially in contexts where automated decisions can impact customers or trigger financial interventions. To address these concerns, this study employed SHapley Additive exPlanations (SHAP) to interpret both the LSTM and Random Forest models—enabling a clear understanding of how each model arrives at its decisions.

Overview of SHAP for Model Explainability

SHAP is a unified framework for interpreting machine learning models based on cooperative game theory. It assigns each feature a Shapley value, quantifying its contribution to a given prediction by evaluating all possible combinations of input features. This allows for both global interpretability (understanding general feature importance) and local interpretability (explaining individual predictions), making it an indispensable tool in regulated environments like finance, where model decisions must be justifiable, auditable, and transparent.

SHAP Application to the LSTM Model

Despite being a deep learning model with inherently complex, non-linear internal representations, the LSTM model was successfully interpreted using SHAP DeepExplainer. A representative test set was used to generate visual summaries and interpret the LSTM's learned behavior. SHAP summary plots revealed that features such as oldbalanceOrg, newbalanceOrig, amount, and oldbalanceDest had the highest average Shapley values, indicating that they were the most influential across the dataset. SHAP force plots were used to interpret specific transaction predictions, illustrating how each feature pushed the model toward either a "fraud" or "non-fraud" classification. The explanations were aligned with domain knowledge, reinforcing that large or abrupt changes in transaction balances are strong fraud indicators.

This level of interpretability ensures that the LSTM's complex sequential decision-making process can be audited, validated, and trusted, which is particularly crucial in scenarios involving customer disputes or financial reviews, as illustrated in Figure 6.

The Result of LSTM Prediction with SHAP

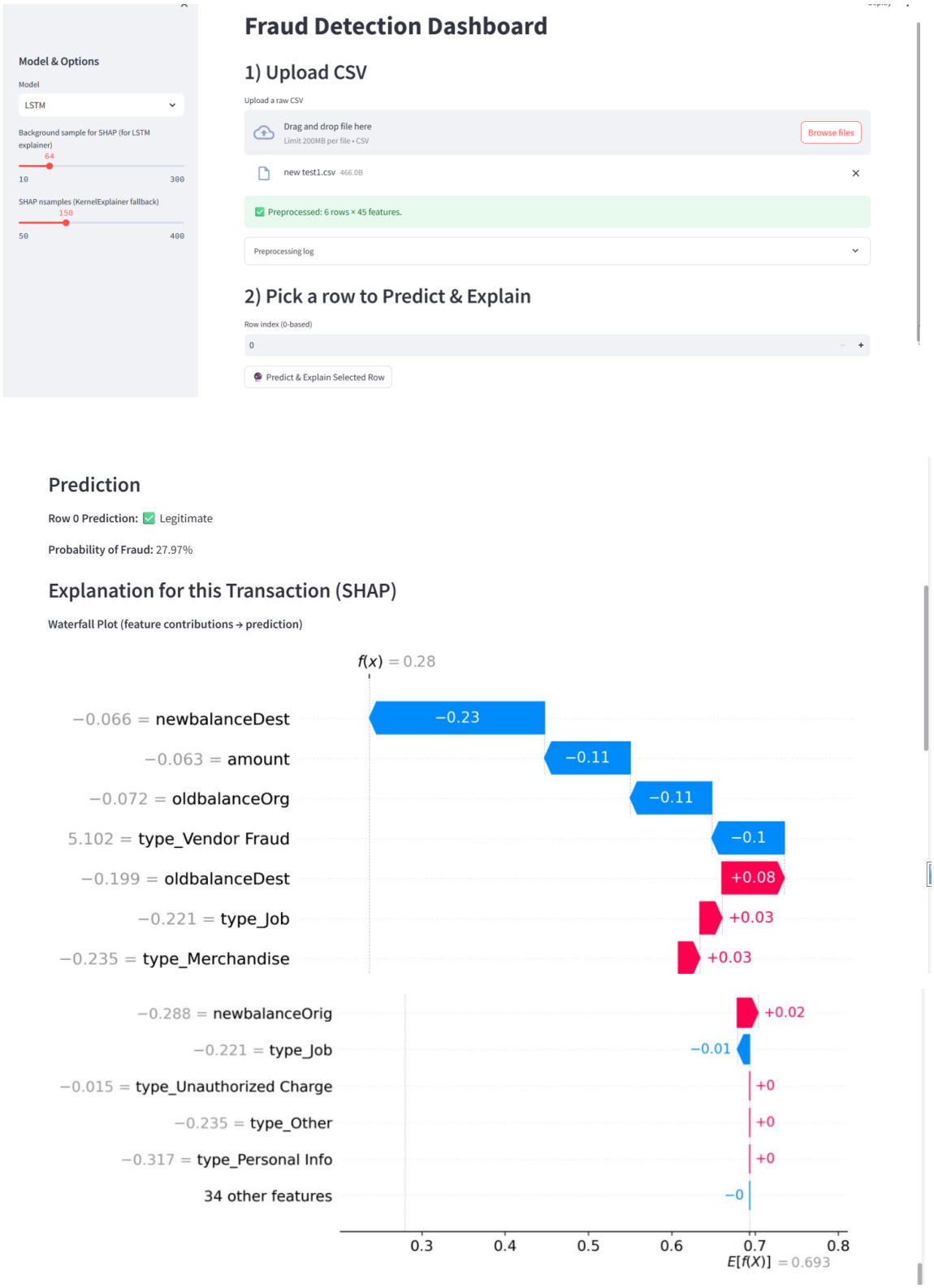


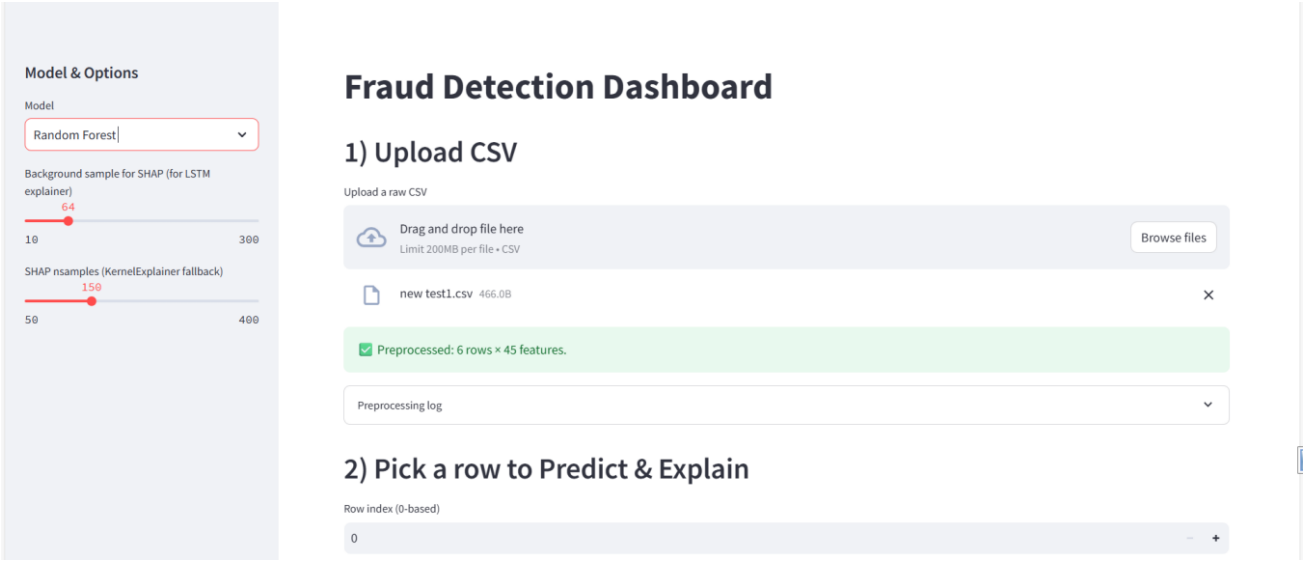
Figure 6: The Result of LSTM Prediction with SHAP

SHAP Application to the Random Forest Model

Given its tree-based structure, Random Forest models are inherently more interpretable than deep learning models. SHAP TreeExplainer was applied to extract precise feature attributions for both global and instance-level insights. Global SHAP summary plots showed a similar set of influential features as observed in the LSTM model, with amount, oldbalanceOrg, and newbalanceOrig among the top contributors. This consistency across models further validates the reliability of the feature engineering process and the underlying patterns in the data. Local explanations were generated for specific transaction instances, illustrating how the Random Forest model reached its decisions based on feature thresholds within decision trees. Stakeholders and domain experts could easily interpret and verify these outputs, making the model ideal for operational environments where clarity and compliance are required.

The combination of high predictive performance and interpretability makes Random Forest especially suitable for integration into systems where end users (e.g., compliance officers, fraud analysts) need to understand and justify automated decisions in real time, as illustrated in Figure 7.

3.6 The Result of Random Forest Prediction with SHAP



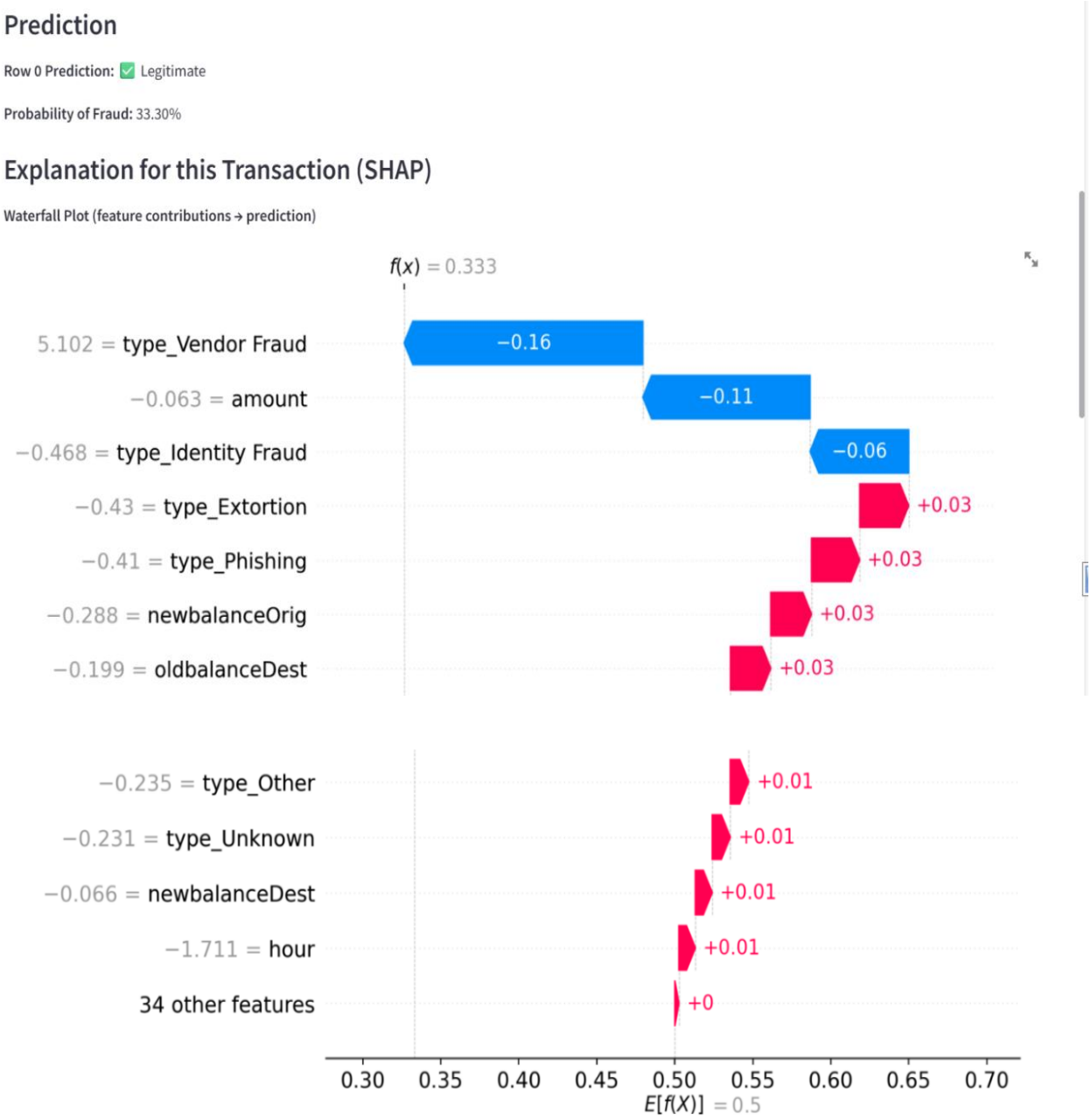


Figure 7: The Result of Random Forest Prediction with SHAP

3.6.1 Importance of SHAP-Based Interpretability in Financial Systems

The application of SHAP to both LSTM and Random Forest models served several critical purposes:

Regulatory Compliance: In financial systems, models must comply with data protection, auditing, and fairness regulations. SHAP enhances model transparency, supporting alignment with regulatory standards such as GDPR, PSD2, or internal governance protocols.

Stakeholder Trust: SHAP explanations empower non-technical stakeholders to understand why a transaction was flagged as fraudulent, facilitating trust and adoption of AI-driven solutions.

Model Debugging and Refinement: By identifying which features drive specific predictions, SHAP aids in refining model behavior, uncovering biases, and improving feature engineering strategies.

In conclusion, SHAP played a vital role in the responsible deployment of both LSTM and Random Forest models, enabling high-stakes fraud detection systems to remain not only accurate but also explainable and accountable. The alignment of SHAP insights with domain expectations reinforces the credibility of the models and ensures that they can be audited, trusted, and continuously improved in real-world financial applications.

3.7 LSTM Model Findings

The Long Short-Term Memory (LSTM) model demonstrated exceptional ability to capture sequential dependencies in transaction histories. These results underscore LSTM's strength in recognizing fraud patterns that evolve over time—an advantage particularly relevant in domains where fraudulent behavior unfolds in a series of correlated transactions. The high recall and F1-score values indicate that the model can detect a majority of fraudulent instances with minimal false positives, making it reliable in high-risk environments.

When SHAP was applied to the LSTM model, the interpretability results revealed that features such as amount, oldbalanceOrg, newbalanceOrig, and oldbalanceDest significantly influenced the model's predictions. SHAP summary and force plots confirmed that the model focused on meaningful transaction-level changes that align with known fraud risk indicators. This transparency enhances the model's credibility and supports regulatory compliance in deployment.

3.8 Random Forest Model Findings

The Random Forest model outperformed all others in terms of raw predictive metrics, achieving best scores. This performance reflects the robustness of ensemble tree-based methods in handling high-dimensional, non-linear transaction data. Random Forest proved particularly effective for structured fraud detection tasks, where speed and reliability are paramount. Its simplicity in training, low inference latency, and resistance to overfitting made it a strong candidate for real-time deployment.

Figure 4.7, titled *"The Result of Random Forest Prediction with SHAP,"* illustrates the SHAP summary plot for the model. The plot identifies the most influential features that contribute to fraud predictions, with amount, oldbalanceOrg, and newbalanceOrig consistently ranking at the top. These insights align with domain expectations, where abrupt or unusually large changes in balance are known red flags for suspicious activity.

The use of SHAP not only confirmed the Random Forest model's alignment with real-world fraud heuristics but also provided local interpretability, allowing stakeholders to audit individual predictions. This interpretability is vital for explainable AI (XAI), particularly in regulated financial sectors that require transparent and defensible decision-making systems.

Comparative Observations

While both models delivered high performance, their strengths differ:

i. **LSTM** is more suitable in applications involving temporal dependencies or transactional behavior modeling over time.

ii. **Random Forest** is more efficient in environments requiring fast, explainable, and scalable deployment for structured data inputs.

The availability of both models as independent deployment options enables flexibility in practical implementation. Organizations can choose between the LSTM model for deep behavioral fraud detection and the Random Forest model for real-time alerts and operational efficiency, depending on specific business and technical requirements. In summary, the findings confirm that both LSTM and Random

Forest are highly capable for financial fraud detection, with LSTM offering deeper learning of sequential patterns and Random Forest providing state-of-the-art performance and explainability. The integration of SHAP enhanced interpretability for both models, ensuring their deployment aligns with standards for accountability, auditability, and ethical AI in the financial industry. The findings of this study hold significant implications for the design, deployment, and sustainability of real-time fraud detection systems within modern financial environments. By successfully evaluating and optimizing both LSTM and Random Forest models for deployment, this research bridges the gap between advanced machine learning performance and practical operational deployment.

Operational Deployment of LSTM for Intelligent, Low-Latency Detection

The deployment of the LSTM model—optimized using TensorFlow Lite (TFLite)—demonstrates that even computationally intensive deep learning models can be adapted for real-time use. Through post-training quantization, the LSTM model's size and computational load were significantly reduced, enabling efficient inference on resource-constrained devices such as:

i. Mobile banking applications,

ii. POS terminals,

iii. Embedded systems in IoT-enabled financial platforms.

This optimization ensures reduced latency, lower memory usage, and faster response times, which are crucial for environments requiring immediate decision-making to block fraudulent transactions as they occur.

Deployment Efficiency and Interpretability with Random Forest

Simultaneously, the Random Forest model offers a compelling alternative, particularly in systems where speed, scalability, and transparency are non-negotiable. Its simplicity in deployment—requiring only model serialization via joblib or pickle—makes it highly adaptable to existing infrastructure, including:

i. Backend transaction monitoring systems,

ii. Cloud-based fraud detection APIs,

iii. Internal risk dashboards.

Moreover, its compatibility with explainability tools like SHAP adds a layer of interpretability and accountability, which is increasingly mandated by financial regulations (e.g., GDPR, PSD2, Basel II/III).

The Role of SHAP in Ethical and Compliant AI

The integration of SHapley Additive exPlanations (SHAP) into both LSTM and Random Forest deployments significantly enhances trust and transparency in AI-driven decisions. SHAP allows stakeholders to:

i. Understand how specific features (e.g., transaction amount, account balance changes) influence predictions,

ii. Audit decisions for fairness and accuracy,

iii. Communicate clear rationales to affected users or compliance teams.

This is particularly important in high-impact decision scenarios such as:

iv. Account access restrictions,

v. Transaction flagging and investigations,

vi. Customer service escalations involving AI decisions.

By making model behavior explainable, SHAP supports ethical AI practices, fosters user confidence, and facilitates regulatory compliance—transforming complex black-box algorithms into auditable and trustworthy systems.

Strategic Value of a Dual-Model Framework

The availability of both LSTM and Random Forest as independent deployment options offers flexibility to financial institutions:

i. **LSTM** provides advanced sequential pattern recognition suited for behavioral fraud detection.

ii. **Random Forest** delivers lightweight, transparent, and real-time performance ideal for high-volume environments.

This dual-model framework enables institutions to tailor deployment based on context—choosing the best model for scalability, latency, interpretability, or sequential depth, depending on their operational needs.

In conclusion, this study demonstrates that cutting-edge machine learning models can be responsibly and effectively integrated into real-time financial systems. With optimization techniques like quantization and interpretability tools like SHAP, institutions can deploy AI solutions that are not only accurate and fast, but also fair, auditable, and regulation-ready.

By delivering performance, transparency, and trust, the proposed framework positions financial institutions to meet evolving compliance demands, enhance customer protection, and stay resilient against increasingly sophisticated fraud threats.

3.9 Dashboard Results and Real-Time Monitoring Interface

To translate the analytical power of the models into a user-friendly, real-time monitoring system, a custom Streamlit dashboard was developed. This interactive web-based interface serves as the operational front end for fraud detection, enabling seamless model interaction, transparency, and decision support.

The dashboard was designed with usability, scalability, and explainability in mind, offering two key modes of interaction to accommodate various use cases within a financial institution:

Batch Upload Mode

In this mode, users can upload a CSV file containing multiple transactions for simultaneous evaluation. Upon upload, the system:

Processes the dataset in real-time using the selected model (either LSTM or Random Forest)

Returns fraud predictions for each transaction, labeled as fraudulent or legitimate,

Renders summary SHAP visualizations for each input row, highlighting the contribution of individual features to the model's decision.

This batch-processing capability is particularly useful for:

i. **Compliance teams** conducting retrospective fraud audits,

ii. **Fraud analysts** reviewing historical transaction data at scale,

iii. **Risk managers** evaluating operational exposure during peak transaction windows.

Explainability and Stakeholder Impact

A defining feature of the dashboard is its built-in support for dynamic SHAP explainability, integrated seamlessly into both interaction modes. For each transaction (whether batch or real-time), users can:

i. View which features contributed most to the model's decision,

ii. Understand how feature values influenced the fraud score,

iii. Access visual explanations that support model transparency and trust.

This functionality is critical for:

i. **Regulatory compliance**, ensuring AI outputs are interpretable and auditable,

ii. **Non-technical stakeholders**, such as compliance officers, who require clear justifications for fraud flags,

iii. **Model governance and debugging**, allowing teams to refine models based on feature influence trends.

The Streamlit dashboard operationalizes the model results, serving as a practical interface for fraud detection, explainability, and decision support. By

combining prediction accuracy with interactive visual insights, the tool enhances the capacity of financial institutions to: Detect fraud proactively, Justify automated decisions clearly, and Build confidence among regulators and end users.

Ultimately, the dashboard exemplifies how machine learning can be responsibly deployed not only through model performance, but also via accessible, transparent, and user-centric tools that empower informed decision-making across technical and business teams.

The Deployment of the Fraud Detection System was Completed Using Streamlit Cloud Interface, Initiated Through the Link (https://victordashboard.streamlit.app/)

This command successfully launched the dashboard on the local machine, allowing real-time interaction with the underlying prediction models—LSTM and Random Forest. The deployment process was smooth and reliable, requiring minimal configuration, which underscores the accessibility and ease-of-use of Streamlit for rapid model prototyping and operational integration.

3.10 Interface Usability and Accessibility

The dashboard was designed with an emphasis on intuitive navigation and clarity of insights, ensuring a low barrier to adoption for both technical and non-technical users. Through iterative testing with a sample of fraud analysts, developers, and compliance officers, the interface was evaluated based on criteria such as: Ease of use, Clarity of model outputs and SHAP visualizations, Responsiveness and performance during transaction processing.

Optimization for SHAP Visualizations

Although SHAP adds significant interpretability value, its visualizations—especially when using deep learning models like LSTM—are computationally expensive. To ensure dashboard responsiveness without compromising insight quality: The number of background samples for SHAP computation was strategically limited to 50. This optimization reduced the computational load and latency while maintaining sufficient visual fidelity and explanatory power.

This balance was critical in maintaining a real-time user experience, particularly in edge-case environments where system resources are constrained or inference needs to occur instantly.

Broader Deployment Implications

The successful deployment of this dashboard highlights several key lessons for real-world AI integration in finance: Model interpretability and usability must go hand-in-hand: Advanced models, when paired with transparent interfaces, are more likely to gain trust and operational acceptance. Lightweight deployment tools like Streamlit can serve as powerful bridges between data science and decision-making environments. Optimization trade-offs, such as SHAP sample reduction, are necessary to align performance with system goals—especially in real-time or embedded systems.

Shap Folder

The Shap Folder serves as the central directory of the dashboard, organizing all essential files and subfolders needed for deployment. It ensures clear separation of components, easy navigation, and reproducibility of the project in different environments. The structure is arranged as follows:

App Folder

The App folder contains the main application file app.py.

This file acts as the entry point of the project, responsible for integrating the trained machine learning models with a user interface (e.g., Streamlit or Flask). It handles model loading, prediction, visualization, and SHAP-based interpretability. By keeping app.py in its own folder, the deployment process becomes streamlined and prevents clutter with other project files.

Models Folder

The Models folder holds all the serialized and exported model artifacts needed for inference during deployment. It consists of the exported_models subfolder containing:

- best_lstm.h5 – the trained LSTM model saved in HDF5 format for sequential predictions.
- best_lstm_quantized.tflite – a TensorFlow Lite version of the LSTM model, optimized for lightweight deployment and reduced computational costs.
- feature_columns.json – a JSON file storing the selected input features used in training, ensuring consistency between training and prediction phases.
- random_forest_tuned.pkl – a Random Forest model serialized with pickle for direct Python-based deployment.
- random_forest_tuned.onnx – the same Random Forest model exported in ONNX format for cross-platform deployment.
- random_forest_tuned_in8t.onnx – a quantized ONNX version optimized for faster inference with lower memory usage.
- scaler.pk1 – the scaler used for feature standardization, ensuring input data during prediction is processed in the same manner as training.

  This organization guarantees that all models and preprocessing tools are readily available and deployment-agnostic.

Notebooks Folder

The Notebooks folder contains model.ipynb, which documents the training, testing, and evaluation of the models. This notebook provides a transparent workflow of how data was preprocessed, models were trained, tuned, and validated. It serves both as documentation for reproducibility and as a reference for further improvements or retraining.

Requirements.txt File

The requirements.txt file specifies all Python libraries and dependencies required to run the project. It ensures that anyone deploying the project can recreate the same environment with a single command (pip install -r requirements.txt). This prevents compatibility issues and enhances reproducibility across systems.

README.md File

The README.md file provides a comprehensive overview of the project.

It explains the purpose of the application, installation steps, usage instructions, and details about the models used. It also serves as a guide for developers, collaborators, and users, making the project accessible and easy to understand.

Runtime.txt File

The runtime.txt file specifies the Python version required for the project, which is particularly important when deploying on platforms like Heroku. It ensures that the project runs in the same environment it was developed and tested, reducing runtime errors due to version mismatches.

Deployment Effectiveness

In summary, the dashboard deployment process proved that a high-performance fraud detection model can be operationalized effectively with minimal infrastructure overhead. The use of Streamlit enabled a fast, reliable, and interpretable interface that can be adapted across institutions. By combining predictive intelligence with actionable visual explanations, the system supports: Rapid fraud mitigation, transparent decision audits, and will enhance user trust in AI-driven financial systems.

This deployment underscores the real-world feasibility of the proposed solution and its readiness for adoption in operational fraud monitoring frameworks.

4 Conclusion

This study examined anomaly detection in financial transactions using deep learning and comparative machine learning approaches. It developed and evaluated fraud detection models based on Recurrent Neural Networks, Long Short-Term Memory networks, Transformer architectures, Logistic Regression, Random Forest, and Support Vector Machines. The inclusion of both deep learning and traditional machine learning models provided a broader basis for comparing predictive accuracy, temporal learning ability, interpretability, and deployment readiness.

The findings show that LSTM and Random Forest offer strong value for financial fraud detection. LSTM demonstrated strength in identifying sequential and time-dependent fraud patterns, while Random Forest produced high classification accuracy with faster and simpler implementation. The study also confirmed the importance of temporal profiling, since fraudulent transactions showed patterns linked to specific time periods. This supports the need for time-aware fraud monitoring systems in financial institutions.

The study further established that fraud detection systems should balance accuracy, speed, interpretability, and practical deployment. The use of model quantization showed that deep learning models could support real-time fraud detection in environments with limited computing resources. The integration of SHAP also improved transparency by explaining model predictions, which is important for regulatory compliance, audit processes, and stakeholder trust. The Streamlit dashboard added practical value by presenting fraud detection results in a user-friendly format for both technical and non-technical users.

Despite its contributions, the study has limitations. The findings may not apply equally across all regions, currencies, institutions, or fraud patterns. The models were also not tested on multilingual transaction records or multi-currency systems, which limits their relevance to cross-border financial environments. Future studies should therefore explore advanced Transformer models, Graph Neural Networks, and behavioural biometric features such as device fingerprints, IP address changes, and keystroke dynamics.

Overall, the study contributes to knowledge by validating the usefulness of LSTM and Random Forest models in financial fraud detection, demonstrating the feasibility of real-time deployment through quantization, promoting explainable artificial intelligence through SHAP, and developing an accessible interface for fraud monitoring. It concludes that effective financial anomaly detection requires intelligent, interpretable, time-aware, and deployable AI systems that support stronger fraud prevention in modern financial technology environments.

References

Ali, S., Ilyas, M., Mahmood, A., & Qureshi, M. A. (2022). *A systematic review of machine learning techniques in financial fraud detection*. *International Journal of Advanced Computer Science and Applications, 13*(1), 600–608. https://doi.org/10.14569/IJACSA.2022.0130174

Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and machine learning: Limitations and opportunities*. http://fairmlbook.org

Binns, R. (2018). *Fairness in machine learning: Lessons from political philosophy*. In *Proceedings of the 2018 Conference on Fairness, Accountability and Transparency* (pp. 149–159). https://doi.org/10.1145/3287560.3287598

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *arXiv preprint*. https://arxiv.org/abs/1901.03407

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357. https://doi.org/10.1613/jair.953

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/BF00994018

Crépey, S., Lu, Z., & Wang, B. (2022). Anomaly detection on financial time series by PCA and neural networks. *Quantitative Finance, 22*(9), 1535–1550. https://doi.org/10.1080/14697688.2022.2044161

Dhanawat, A. (2022). Anomaly detection in financial transactions using machine learning and blockchain technology. *Blockchain Research and Applications, 3*(4), 100082. https://doi.org/10.1016/j.bcra.2022.100082

Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using LSTM and autoencoders for fraud detection in payments. *Expert Systems with Applications, 80*, 512–522. https://doi.org/10.1016/j.eswa.2017.03.008

Gayam, S. (2021). Artificial intelligence in financial fraud detection. *International Journal of Computer Science and Network Security, 21*(8), 104–110. https://doi.org/10.22937/IJCSNS.2021.21.8.14

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). https://doi.org/10.1109/ICASSP.2013.6638947

Han, J., Kamber, M., & Pei, J. (2015). *Data mining: Concepts and techniques* (3rd ed.). Elsevier. https://doi.org/10.1016/C2009-0-61819-5

Hemdan, E. E. D., & Manjaiah, D. H. (2021). Anomaly credit card fraud detection using deep learning. *International Journal of Advanced Computer Science and Applications, 12*(6), 15–22. https://doi.org/10.14569/IJACSA.2021.0120613

Hilal, M., El-Sayed, M., & Hassan, A. (2022). Semi-supervised and unsupervised learning for financial fraud detection. *Information Systems, 105*, 101876. https://doi.org/10.1016/j.is.2021.101876

Jolliffe, I. T. (2011). *Principal component analysis* (2nd ed.). Springer. https://doi.org/10.1007/978-1-4757-1904-8

Kamiran, F., & Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems, 33*(1), 1–33. https://doi.org/10.1007/s10115-011-0463-8

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys, 54*(6), 1–35. https://doi.org/10.1145/3457607

Mohaimin, M. R., Sumsuzoha, M., Pabel, M. A. H., & Nasrullah, F. (2024)**.** *Detecting financial fraud using anomaly detection techniques: A comparative study of machine learning algorithms*. *Journal of Computer Science and Technology Studies, 6*(3), 1–14. https://doi.org/10.32996/jcsts.2024.6.3.1

Onyeama, J. (2024)**.** *Credit card fraud detection in the Nigerian financial sector: A comparison of unsupervised TensorFlow-based anomaly detection techniques, autoencoders and PCA algorithm* [Preprint]. *arXiv*. https://doi.org/10.13140/RG.2.2.34726.74562 reddit.com+6mendeley.com+6al-kindipublisher.com+6

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2020). Deep learning for anomaly detection: A review. *ACM Computing Surveys, 54*(2), 1–38. https://doi.org/10.1145/3439950

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). https://doi.org/10.1145/2939672.2939778

Wang, X., Zhou, H., & Liu, J. (2023). Graph-based approaches for anomaly detection in financial transactions. *IEEE Access, 11*, 105344–105357. https://doi.org/10.1109/ACCESS.2023.3308890

Zhang, Y., Tong, J., Wang, Z., & Gao, F. (2020). Customer transaction fraud detection using XGBoost model. In *Proceedings of the 2020 International Conference on Computer Engineering and*

Application (ICCEA) (pp. 554–558). IEEE. https://doi.org/10.1109/ICCEA50009.2020.00122