TRANSACTIONAL FRAUD DETECTION USING PYTHON CLASSIFICATION MODELS FOR A U.S. RETAIL BANK

1. Background

A mid-sized U.S. retail bank offering credit and debit card services across 10 states faced a growing problem of undetected low-value fraud cases. The internal system relied on rule-based logic, which was missing evolving fraud patterns and generating too many false alerts.

We were brought in to develop a machine learning–based fraud detection model using Python. The aim was to uncover hidden behavioral patterns, reduce false positives, and prioritize high-risk transactions for review, all while integrating smoothly into the client's analytics stack.

2. Objective

- To identify key predictors of fraudulent activity from transaction logs
- To build binary classification models using Python that detect fraud with high precision and recall
- To evaluate and compare model performance (Logistic Regression, Random Forest, Gradient Boosting)
- To generate interpretable output and risk scores for integration into the bank's internal fraud dashboard

3. Data Used

Source: Internal anonymized transaction logs (6-month window)

Dataset Details:

- ~2.1 million transactions
- Fields:
  - Transaction_ID, Customer_ID, Amount, Merchant_Category, Transaction_Type
  - Time_of_Day, Geo_Location, Device_Type, Channel (Online, POS, ATM)
  - Is_Fraud (target variable)

Fraud Rate: ~0.38% (high class imbalance)

4. Methodology

4.1 Data Preparation

- Balanced dataset using SMOTE (Synthetic Minority Over-sampling Technique)
- Created derived features:
  - Transaction_Amount_Deviation
  - Customer_Historical_Fraud_Count
  - Time_Since_Last_Transaction
- Handled missing device metadata using median imputation

4.2 Model Building

- Models tested:
  - **Logistic Regression** (baseline, interpretable)
  - **Random Forest** (strong classifier, low tuning needed)
  - **XGBoost** (high-performance, tuned using GridSearchCV)
- Evaluation metrics:
  - Precision, Recall, F1-score, ROC AUC
  - Confusion Matrix and Cost-based Analysis
- Libraries: pandas, scikit-learn, xgboost, imbalanced-learn, matplotlib, seaborn

5. Mining Results

Model	Precision	Recall	F1 Score	ROC AUC
Logistic Regression	0.76	0.55	0.64	0.88
Random Forest	0.85	0.73	0.78	0.95
XGBoost	0.89	0.81	0.85	0.97

- **XGBoost outperformed** other models on all key metrics
- Most predictive features:
  - Transaction_Amount_Deviation
  - Channel (Online fraud > POS fraud)

- Transaction_Time_Bin (late-night spikes in fraud risk)

6. Strategic Insights

- 90% of fraud cases were captured **before human review** using top-scoring transactions
- Recommended real-time flagging system for high-risk patterns:
  - Large deviation in amount + foreign device + non-home location
- Suggested segmentation of fraud rules by **channel type** to reduce false alarms in POS transactions
- Provided explainable decision trees for audit teams to review flagged alerts with confidence

7. Reporting Output

- **Python Script (Jupyter Notebook)**:
  - End-to-end fraud detection pipeline
  - Hyperparameter tuning grid
  - Model export (.pkl) for production deployment
- **PDF Report (20 pages)**:
  - Executive overview
  - Model comparison and confusion matrices
  - Top fraud patterns and visual behavior trees
- **Excel Output**:
  - Transaction risk scores
  - Top 1,000 highest-risk transactions with decision logic trail
  - Threshold tuning tool for alert calibration

8. Business Impact

- **False positives reduced by 32%**, freeing internal fraud analysts for serious alerts
- Detected 9 of 10 fraud rings operating under previously unseen transaction patterns

- Risk scoring logic integrated into the bank's fraud monitoring dashboard via scheduled script
- Model now updated monthly using rolling training data and re-validated quarterly