Project Report: Proactive Fraud Detection Using Machine Learning

---

1. Introduction

Fraud detection is a critical challenge for financial institutions due to the significant financial and reputational loss it causes. In this project, we develop a machine learning-based pipeline to proactively detect fraudulent financial transactions. We used a realistic synthetic dataset and applied multiple models and evaluation techniques to build a robust fraud detection system.

---

2. Data Overview

- The dataset contains millions of transactions, including TRANSFER, CASH_OUT, PAYMENT, etc.
- The target variable is isFraud (1 = fraud, 0 = legitimate).
- No missing values were found in the dataset.
- Data is available in CSV format having 6362620 rows and 10 columns.

---

3. Data Cleaning and Preparation

- **Missing Values:** Checked using .isnull().sum() — no missing values found.
- **Outliers:** Retained in the dataset, as frauds are inherently outliers. Instead, captured suspicious behavior using flags like:
    - high_amount_flag (amount > 90% of balance)
    - same_amount_flag (consecutive same amounts)
    - transfer_cashout_block (fraud-like sequences)
- **Multicollinearity:**
    - Redundant balance-related columns were dropped after creating more meaningful features (e.g., balance_check, high_amount_flag).
    - Manual inspection and correlation heatmaps helped in selecting non-redundant features.

---

4. Feature Engineering

- same_amount_flag: Flags transactions with repeated consecutive amounts, indicating automation or bot-like behavior.
- transfer_cashout_block: Detects a pattern where a TRANSFER is immediately followed by a CASH_OUT of the same amount. All frauds occurred in this pattern.
- high_amount_flag: Flags transactions where the amount is greater than 90% of the account balance.
- One-hot encoding applied to type variable (TRANSFER, CASH_OUT highly predictive).

---

5. Modeling and Evaluation

Models Tried:

- Logistic Regression: Baseline, interpretable.
- Random Forest: High performance, interpretable feature importance.
- XGBoost: Best performance, robust to imbalance.

Imbalance Handling:

- Used class_weight='balanced' for tree models.
- Sampling for train and test sets was stratified.

Evaluation Metrics:

- Confusion Matrix
- Accuracy, Precision, Recall, F1 Score
- Plotted F1 Score comparison across models

Best Model: Logistic Regression

- Accuracy: **1.0000**
- Precision: **1.0000**
- Recall: **0.9957**
- F1 Score: **0.9979**

---

6. Feature Importance and Interpretation

- transfer_cashout_block: Strongest indicator of fraud. All frauds occurred in this block.
- same_amount_flag: Indicates automation or repeated fraud attempts.
- type: Only TRANSFER and CASH_OUT are linked to fraud.
- high_amount_flag: Indicates intent to drain accounts.

These factors align well with real-world fraud behaviors:

- Fraudsters often chain transfers and cash outs.
- They frequently use identical amounts for consistency or bot automation.
- Large withdrawals are often attempted in one go.
- The pattern-based features captured hidden fraud indicators effectively.

---

7. Fraud Prevention Recommendations

- Monitor transfer-to-cashout sequences in real-time.
- Apply rate limits and stricter verification for large withdrawals.
- Introduce adaptive learning pipelines to retrain models periodically.
- Use behavioral features like repeated amounts to flag accounts.

---

8. A/B Testing to Validate Improvements

Method:

- Split users or transactions into two groups:
  - Group A (control): Current fraud detection system
  - Group B (test): Updated system with new models and flags

Evaluate:

- Compare fraud metrics (false negatives, recall) before and after implementation.
- Use A/B testing on transaction pipelines.
- Monitor fraud trend over time.
- Track customer disputes and complaint rates.
- Run periodic model revalidation to measure detection quality.

Decision: Adopt the new system if Group B performs significantly better.

---

9. How Features Were Selected

Selection Strategy:

- Removed identification variables (nameOrig, nameDest, etc.).
- One-hot encoded the type variable.
- Engineered fraud-relevant features based on domain intuition:
    - same_amount_flag
    - transfer_cashout_block
    - high_amount_flag
- Ensured low correlation between remaining features.

---

10. Conclusion

This project demonstrated that behavioral pattern engineering combined with tree-based machine learning models can significantly improve fraud detection. The Logistic Regression model with engineered features showed outstanding performance. A/B testing is recommended for safely validating and rolling out model updates in production environments.