Fraud Detection Analysis

Business Context

This notebook presents a solution for predicting fraudulent transactions. The goal is to develop a machine learning model and provide actionable insights.

Candidate Expectations & Answers

This notebook is structured to address the following:

1. Data Cleaning & EDA
2. Model Description
3. Variable Selection
4. Model Performance
5. Key Predictive Factors
6. Factor Interpretation
7. Prevention Strategies
8. Action Evaluation

1. Data Loading and Imports

2. Data Cleaning and EDA

Question 1: Data cleaning including missing values, outliers and multi-collinearity.

We investigate missing values, check for outliers in numerical columns, and analyze multi-collinearity.

Observations:

- No missing values found in the dataset.
- **Outliers**: The `amount` column has a massive range (max ~92M vs mean ~180k). Similarly for balances. These are likely valid large transactions rather than errors, but they are outliers in the statistical sense.
- **Multi-collinearity**: We expect `oldbalanceOrg` and `newbalanceOrig` to be highly correlated.

3. Feature Engineering and Variable Selection

Question 3: How did you select variables to be included in the model?

Selection Strategy:

1. **Included**: `step`, `type`, `amount`, `oldbalanceOrg`, `newbalanceOrig`, `oldbalanceDest`, `newbalanceDest`. These provide the core transaction details.
2. **Excluded**: `nameOrig`, `nameDest`. High cardinality categorical variables. While specific accounts might be repeat offenders, for a generalizable model, we initially exclude them to avoid overfitting to specific IDs. `isFlaggedFraud` is also excluded to prevent leakage if it's a post-event flag, or it can be used as a baseline comparison.
3. **Feature Engineering**:
   - `type`: Converted to numerical using Label Encoding.
   - `errorBalanceOrig`: Difference between original balance difference and transaction amount.
   - `errorBalanceDest`: Difference between destination balance difference and transaction amount.

4. Model Development

Question 2: Describe your fraud detection model in elaboration.

We use **XGBoost (Extreme Gradient Boosting)**. **Reasons:**

- Handling Large Data: Efficient matrix operations.
- Non-Linearity: Tree-based models capture complex interactions between balance and amount.
- Class Imbalance: XGBoost has built-in `scale_pos_weight` to handle imbalance.
- Interpretability: Provides feature importance scores.

We split the data 80% Training (Calibration) and 20% Testing (Validation).

5. Model Evaluation

Question 4: Demonstrate the performance of the model by using best set of tools.

We evaluate using Precision, Recall, F1-Score, and AUC-ROC.

6. Interpretability and Insights

Question 5: What are the key factors that predict fraudulent customer? Question 6: Do these factors make sense? If yes, How? If not, How not?

Analysis of Factors: The model typically highlights:

1. **errorBalanceOrig/Dest**: Large discrepancies in expected balance updates are strong indicators.
2. **amount**: Fraudulent transactions are often large efforts to cash out.
3. **type**: Transfer and Cash-Out are the primary vectors for fraud.

Do they make sense? Yes. Fraudsters aim to steal money (high amount), often move it (Transfer) and withdraw it (Cash-Out). They might leave the account in an inconsistent state or drain it completely (oldbalanceOrg = amount), leading to specific balance patterns.

7. Actionable Plan

Question 7: What kind of prevention should be adopted while company update its infrastructure?

1. **Real-time Transaction Scoring**: Deploy this XGBoost model to score transactions in real-time. Block or flag transactions with score > threshold (e.g., 0.9).
2. **Velocity Checks**: Limit the number/amount of transactions within a time window (using the `step` variable intuition).
3. **Two-Factor Authentication (2FA)**: Trigger 2FA for high-risk transactions (e.g., Transfers > $10,000).
4. **Balance Validation**: Implement strict ACID compliance and checks to ensure `newbalance` = `oldbalance` - `amount`. Any deviation should auto-freeze.

Question 8: Assuming these actions have been implemented, how would you determine if they work?

1. **A/B Testing**: Run the model in "shadow mode" first, then on a subset of users.
2. **Metric Monitoring**:
   - **Fraud Rate**: Should decrease.
   - **False Positive Rate (Customer Friction)**: Should remain low. If too many legitimate users are blocked, adjust the threshold.
   - **Chargeback Rates**: A lagging indicator that should drop over months.