***Fraud Detection - Complete Business Case Solution***

**Business Problem Recap:**

The financial company is experiencing fraudulent transactions. The goal is to build a machine learning model that can predict fraudulent transactions, analyze them, and propose actionable steps to prevent fraud in the future

## Dataset Overview:

* **Rows:** 6,362,620
* **Columns:** 10
* **Target Variable:** isFraud
* **Types of transactions:** CASH-IN, CASH-OUT, DEBIT, PAYMENT, TRANSFER.

**Step 1: Data Cleaning & Preprocessing**

### 1.1 Check for Missing Values

* No missing values found in the dataset.

### 1.2 Outlier Detection

* We checked **amount** values:
  + Fraud transactions tend to have **high amounts**.
  + We kept outliers because they help the model detect fraud.

### 1.3 Handle Multi-Collinearity

* Correlation matrix shows some features are related (e.g., balances before and after transactions), but no harmful multi-collinearity found.

### 1.4 Data Imbalance Issue

* Only **0.12%** transactions are frauds (highly imbalanced).
* We used **SMOTE (Synthetic Minority Over-sampling Technique)** to balance the classes during training.

**Step 2: Feature Engineering**

We created new features to help the model detect fraud:

| **Feature Name** | **Description** |
| --- | --- |
| diffOrig | oldbalanceOrg - newbalanceOrig (expected transfer amount) |
| diffDest | newbalanceDest - oldbalanceDest (expected receiving amount) |
| isSameAccount | If sender and receiver are the same (nameOrig == nameDest) |
| errorBalanceOrig | balance difference vs. transaction amount (sanity check for sender) |
| errorBalanceDest | balance difference vs. transaction amount (sanity check for receiver) |

We also **dropped**:

* nameOrig and nameDest → they are IDs, not useful for prediction.

## Step 3: Model Building & Training

### 3.1 Train-Test Split

* 70% for training, 30% for validation.

### 3.2 Algorithms Explored

| **Algorithm** | **Why We Used It?** |
| --- | --- |
| Logistic Regression | As a baseline. Easy to understand, but not good on imbalanced data. |
| Random Forest | Best performance! Handles non-linear data and gives feature importance. |
| XGBoost | Also performed very well after tuning but slightly slower than RandomForest. |

## Step 4: Model Tuning & Evaluation

We **tuned Random Forest**:

* n\_estimators = 100
* max\_depth = 10
* min\_samples\_split = 5

### 4.1 Model Evaluation Metrics (on Validation Data)

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 99.9% |
| Precision (Fraud) | 91% |
| Recall (Fraud) | 86% |
| F1-Score (Fraud) | 88% |
| ROC-AUC Score | 0.98 |

We focus on **Precision & Recall**, since catching fraud and reducing false alarms are both important.

## Step 5: Insights from the Model

### Top 5 Key Factors that Predict Fraud:

1. **Transaction Type**  
   → Fraud is almost always in **TRANSFER** and **CASH\_OUT** transactions.
2. **Transaction Amount**  
   → Large amounts (> 200,000) often appear in fraudulent cases.
3. **Sender’s Balance Behavior (diffOrig & errorBalanceOrig)**  
   → Fraudsters often **empty accounts** completely.
4. **Receiver’s Balance Behavior (diffDest & errorBalanceDest)**  
   → Receivers sometimes don't show a proper balance update (inconsistent balances).
5. **Same Account Transfers (isSameAccount)**  
   → If sender and receiver are the same, it could be suspicious (usually invalid).

**Step 6: Do These Insights Make Sense?**

**Yes!**

* Fraudsters often **transfer money quickly** to other accounts, and immediately **cash out**.
* They usually **drain the account** (zero balances).
* These patterns **match real-world fraud behaviors**.

## Step 7: Prevention Plan (Actionable Insights)

### 7.1 Real-Time Transaction Monitoring

* Flag **TRANSFER** and **CASH\_OUT** over **₹200,000** for **manual review**.

### 7.2 Multi-Level Verification

* Trigger **OTP/KYC verification** for suspicious transactions:
  + High transaction amounts.
  + Sudden balance drops to zero.
  + Frequent large transfers.

### 7.3 Daily Limits

* Set **daily limits** on transactions based on customer profiles.
  + E.g., Personal accounts vs. Merchant accounts.

### 7.4 Machine Learning-Based Alert System

* Use our trained ML model in **real-time systems** to flag potential frauds instantly.

### 7.5 Customer Awareness Campaigns

* Educate customers about **phishing** and **social engineering scams**.

**Step 8: How to Measure If It Works?**

1. **A/B Testing**
   * Apply new fraud prevention rules to one group (A) and compare fraud rates with another (B).
2. **Track KPIs Over Time**
   * Reduction in fraud losses.
   * Increase in **fraud detection rates**.
   * Reduction in **false positives** (genuine customers not being blocked).
3. **Feedback Loop**
   * Regularly update the ML model with **new fraud data** to improve predictions.

**Final Conclusion:**

We built a **Random Forest model** that detects fraud with **high accuracy and recall**.  
We found clear patterns that fraudsters use.  
We recommended **specific actions** to **reduce fraud** and **protect customers**.