Fraud Detection Project Report

Introduction

The goal of this project was to build a machine learning model to predict whether a financial transaction is fraudulent. Fraud detection is important because fraudulent transactions, though rare, can cause serious losses. The challenge is that fraud data is highly imbalanced which means most transactions are normal, and only a few are fraudulent.

Dataset

I created a synthetic dataset of 10,000 training and 1,500 test transactions. It included features such as:

- Transaction details: amount, type, channel, merchant category
- Customer information: age, income, tenure, location, email domain
- Account features: balances before/after, average spend, transaction counts
- Device/risk features: device trust score, proxy IP, foreign/high-risk transactions

ID columns (transaction_id, customer_id) were dropped and is_fraud was used as the target.

Data Preprocessing

- Missing values: Replaced with outlier values (e.g., 99999) so the model could learn missingness as a potential fraud signal.
- Encoding: Used OrdinalEncoder for categorical features (pipeline-friendly).
- Scaling: Standardized numerical features for linear models excluding the encoded categories; not required for tree-based models.

Modeling & Class Imbalance Handling

I first explored models using LazyPredict. Then I focused on Logistic Regression, Random Forest, and XGBoost.

Because fraud cases were ~4%, I handled imbalance using class weights (class_weight='balanced'). I chose not to use oversampling since class weights already improved results.

Model Selection & Results

Random Forest and XGBoost showed high accuracy (~0.95) but failed to detect fraud (recall = 0). Logistic Regression with class weights performed better. After tuning, it gave:

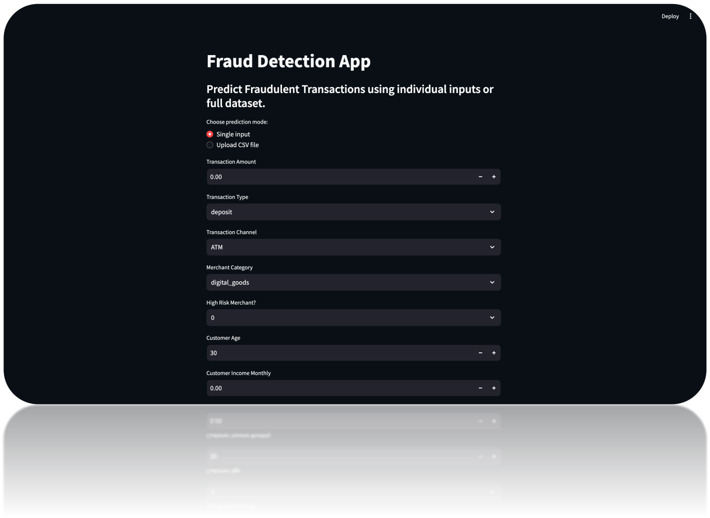
- Accuracy: ~0.78, Balanced Accuracy: ~0.63
- Precision: ~0.10, Recall: ~0.46, F1: ~0.16
- ROC AUC: ~0.64, PR AUC: ~0.08

Although PR AUC looks small, it is about twice the random baseline (~0.04), showing the model captured useful patterns. I also adjusted the threshold (0.45) to improve recall.

Deployment

I deployed the final pipeline with Streamlit. Users can:
- Enter a single transaction manually, or
- Upload a CSV file for batch predictions.

This was used to predict the outcome of the synthetic 1,500 test transactions.



Fraud Detection App

Predict Fraudulent Transactions using individual inputs or full dataset.

Choose prediction mode:
● Single input
○ Upload CSV file

Transaction Amount
0.00

Transaction Type
deposit

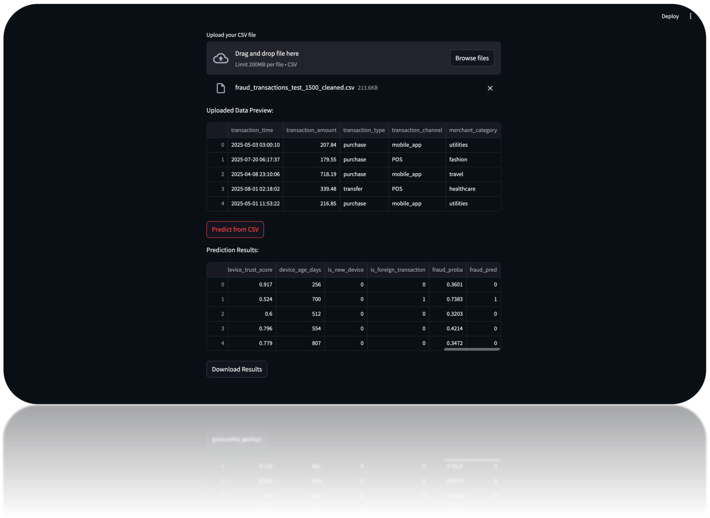
Transaction Channel
ATM

Merchant Category
digital_goods

High Risk Merchant?
0

Customer Age
30

Customer Income Monthly
0.00



Upload your CSV file

Drag and drop file here
Limit 200MB per file • CSV

fraud_transactions_test_1500_cleaned.csv 213.6KB

Uploaded Data Preview:

	transaction_time	transaction_amount	transaction_type	transaction_channel	merchant_category
0	2025-05-03 03:00:10	207.84	purchase	mobile_app	utilities
1	2025-07-20 06:17:37	179.55	purchase	POS	fashion
2	2025-04-08 23:10:06	718.19	purchase	mobile_app	travel
3	2025-08-01 02:18:02	339.48	transfer	POS	healthcare
4	2025-05-01 11:53:22	216.85	purchase	mobile_app	utilities

Predict from CSV

Prediction Results:

	device_trust_score	device_age_days	is_new_device	is_foreign_transaction	fraud_proba	fraud_pred
0	0.917	256	0	0	0.3601	0
1	0.524	700	0	1	0.7383	1
2	0.6	512	0	0	0.3203	0
3	0.796	554	0	0	0.4214	0
4	0.779	807	0	0	0.3472	0

Download Results

Bonus: Data Drift

Fraud patterns can change over time (data drift). To check this, I used the Kolmogorov–Smirnov (KS) test to compare feature distributions between training and test sets. Other common tests include Chi-square (categorical), PSI (finance), and Jensen–Shannon divergence (information theory).

Why These Metrics Were Used

- Accuracy: % correct predictions, but misleading for imbalanced data.
- Balanced Accuracy: Accounts for both classes equally.
- Precision: Of flagged frauds, how many were real?
- Recall: Of actual frauds, how many were caught? (most important).
- F1 Score: Balance between precision and recall.
- ROC AUC: Overall separation between fraud/non-fraud.
- PR AUC: Focuses on fraud (minority class). 0.08 may look low but is meaningful since baseline is ~0.04.

Conclusion

This project demonstrated how to:
1. Create synthetic fraud data and preprocess it.
2. Handle imbalanced data.
3. Train and evaluate ML models.
4. Select Logistic Regression with class weights as the best model.
5. Deploy the model via Streamlit.
6. Understand data drift as a future risk.

This thought me that fraud detection is not about perfect accuracy, it is about maximizing fraud recall while minimizing false alarms.