Proposed Solution

Online Payments Fraud Detection using Machine Learning

1. Proposed Solution Overview

The proposed solution is a full-stack machine learning application consisting of two components:

1. A trained Random Forest classifier serialized as a pickle file (payments.pkl)
2. A Flask web application serving a 3-page interface for transaction analysis

2. ML Pipeline

2.1 Data Preprocessing

- Load PaySim dataset (6.3M rows, 11 columns)
- Encode 'type' column: PAYMENT=0, TRANSFER=1, CASH_OUT=2, DEBIT=3, CASH_IN=4
- Select 7 features: step, type, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest
- 80/20 train-test split with stratification

2.2 Model Training

- Algorithm: RandomForestClassifier (scikit-learn)
- Training samples: ~5.05M, Test samples: ~1.26M
- Hyperparameters: n_estimators=100, default settings (no grid search required)
- Training time: ~15 minutes on standard laptop

2.3 Inference Pipeline

```
User input → Flask POST /submit → numpy array (1,7) → model.predict_proba() → fraud
probability → threshold check (>0.20) → render result
```

3. Web Application Design

Route	Method	Description
/	GET	Home page — project overview and CTA button

Route	Method	Description
/predict	GET	Transaction form — 7-field input form with type selector
/submit	POST	Result page — fraud verdict + probability bar + explanation

4. Key Design Decisions

Threshold = 0.20

The default 0.50 threshold was too conservative, missing real fraud cases. Lowering to 0.20 improves recall at a small cost to precision. This is the correct trade-off for fraud detection where missing fraud is more costly than a false alarm.

Probability Score Display

Rather than only showing a binary verdict, the result page displays the raw probability percentage (e.g., 73.4%). This enables human analysts to apply their own judgment for borderline cases.