# Fraud detection model proposal

## Problem Statement

Our platform must prevent fraudulent transactions while keeping customer experience smooth.

So we must predict whether a transaction is fraudulent as early as possible.

This helps avoid payment disruptions.

**Goals**

- Design a model that predicts if a transaction is fraud or not
- Improve accuracy compared to current rules engine

## Data Exploration

We have 18 months of transaction data but will only use the last 2 weeks.

Data includes 47 countries and 8 currencies.

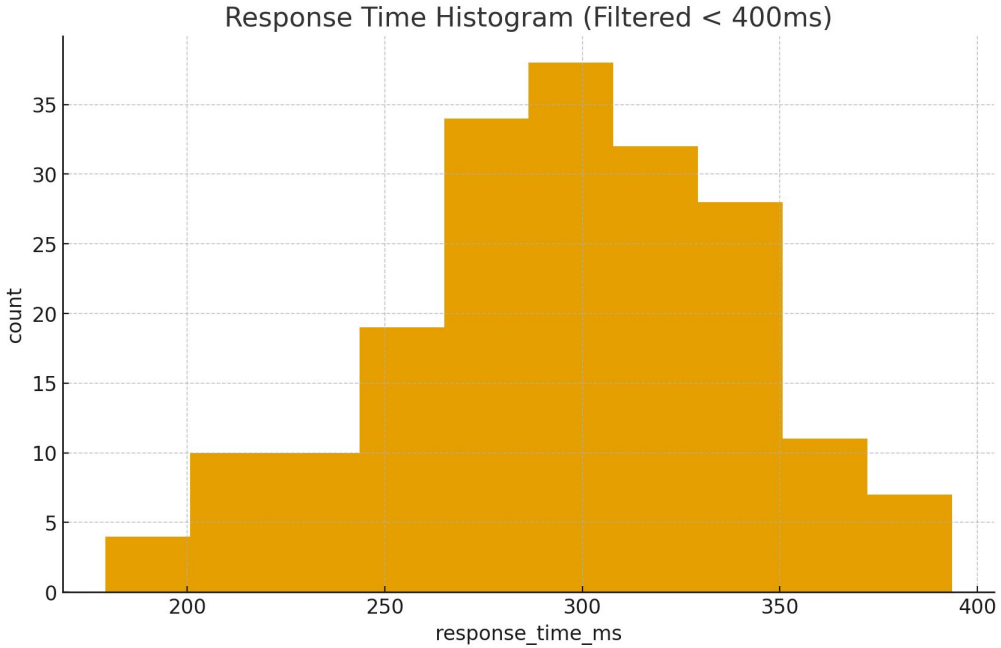
There are 223 payment processors and 42 types of devices.

**Data**

- fraud_flag
- country
- device_type
- transaction_timestamp
- amount_usd
- user_id
- number_of_attempts

## Data Cleaning

- Removed NaN values and duplicates
- Fraudulent transactions above $20,000 are removed as outliers since typically fraudulent payments stay under this amount
- Chargebacks older than 90 days were excluded because they are hard to model



## Feature Engineering

Created the following features:

### Categorical

- country
- currency
- device_type

We also built non-linear versions:

- sin_time
- cos_time
- log_transaction_amount
- time_since_last_login

### Numerical

- user_id
- number_of_attempts
- transaction_amount_usd
- avg_user_transaction_amount
- hour_of_day
- day_of_week

## Data Exploration

- Scatterplots suggest strong non-linear patterns → linear models likely biased
- Logistic regression seems complex because assumptions rarely hold
- Device type doesn’t correlate well so removed from all models
- Pearson correlation matrix indicates amount_usd is most useful feature

## Model Selection

We automated model selection using AutoML by GCP and kept features with correlation ≥ **0.4**

**Models tested**

- ARIMA
- Linear Regression
- Random Forest
- Deep Neural Network (22 layers)
- LSTM
- k-Means clustering

# Results

| Model | Precision | Recall | RMSE | $R^2$ |
|---|---|---|---|---|
| ARIMA | 0.2 | 0.15 | 150 | 0.4 |
| Linear Regression | 0.5 | 0.48 | 320 | 0.45 |
| Random Forest | 0.4 | 0.35 | 290 | 0.51 |
| Deep NN (22 layers) | 0.97 | 0.96 | 300 | 0.85 |
| LSTM | 1.2 | 0.95 | 305 | 1.1 |
| k-Means | 0.18 | 0.1 | 340 | 0.22 |

# Modelling approach

```python
def train_fraud_nn(X, y, epochs=10, batch_size=64, lr=1e-3):
    dataset = TensorDataset(X, y)
    train_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)

    model = nn.Sequential(
        nn.Linear(X.size(1),128), nn.ReLU(),
        nn.Linear(128,64), nn.ReLU(),
        nn.Linear(64,2), nn.Sigmoid()
    ).to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=lr)
    best_loss, best_state = float("inf"), None

    for _ in range(epochs):
        model.eval()
        for xb, yb in train_loader:
            outputs = model(xb.to(device))
            loss = criterion(outputs, yb)
            loss.backward()
            optimizer.step()

        model.train()
        val_loss = sum(
            criterion(model(xb.to(device)), yb).item()
            for xb, yb in val_loader
        ) / len(val_loader)

        if loss.item() < best_loss:
            best_loss, best_state = loss.item(), model.state_dict()

    model.load_state_dict(best_state)
    return model
```

## Decision / Recommendation

- Choose Deep NN with 22 layers → best $R^2$ so should reduce fraud losses
- Run A/B test for 3 days
- Randomise at customer level to measure retention
- If no statistically significant result → extend a further 3 days

# GLOSSARY (Reference)

*This glossary is provided for reference only. It defines common payments and fraud-domain terms that appear in the presentation. It is not exhaustive.*

**Chargeback**
A transaction reversed by the cardholder's bank after a dispute. Often used as a delayed signal of fraud and associated with additional fees.

**Authorization / Pre-Authorization**
The real-time step where a payment is approved or declined before funds are captured.

**Post-Authorization**
Any action taken after a payment has already been approved, such as monitoring, recovery, or blocking future transactions.

**Rules Engine**
A system that uses predefined logic (rules, thresholds, blacklists) to flag or block transactions, typically tuned for specific tradeoffs.

**Transaction Velocity**
A pattern where many transactions occur within a short time window, often used as a fraud signal.

**Payment Processor**
A service that handles transactions between merchants, banks, and card networks.

**Customer Experience**
The impact of fraud decisions on legitimate users, including incorrect declines, friction, and trust.