**Objective**  
Detect fraudulent transactions under severe class imbalance while minimizing false positives that impact customers.

**Technologies**  
Python, pandas/NumPy, scikit‑learn (**Logistic Regression, Random Forest**), **XGBoost**; class weighting and decision‑threshold sweeps; matplotlib for PR/ROC.

**Role / Contributions**

* Led data cleaning, EDA, and **leakage‑safe** feature engineering: Haversine distance (cardholder↔merchant), category/merchant risk encodings, velocity features.
* Designed a **four‑model progression** to isolate value from domain features:

• **Model 1 — Logistic Regression (baseline):** clean tabular features to set reference metrics.  
• **Model 2 — Random Forest + Distance:** added Haversine distance to capture implausible travel.  
• **Model 3 — XGBoost + Category Risk:** injected train‑only category fraud‑rate encodings.  
• **Model 4 — Random Forest + Merchant Risk + Velocity:** layered merchant risk and burst‑activity features for cash‑out patterns.

* Handled imbalance via **class weights** and **threshold tuning**; published PR/ROC curves, confusion matrices, and a short **threshold policy** for stakeholders.

**Outcomes / Results**

Clear lift across models:

• **Model 1 (LR baseline):** AUC ~0.93; Precision/Recall ≈ 45%/47%.  
• **Model 2 (RF + distance):** AUC ≈ 0.981; P/R ≈ 74%/72%.  
• **Model 3 (XGB + category risk):** AUC ≈ 0.980; P/R ≈ 85%/65% (high‑precision option).  
• **Model 4 (RF + merchant risk + velocity):** AUC ≈ 0.990; P/R ≈ 79%/75% (best overall balance).