

AI Fraud Detection — Product Document

Oct 3, 2026 · @Harshal Borole

Executive summary

FraudShield is an AI decision-support layer that scores every bank transaction for fraud risk in under 50 ms. It plugs into a legacy core banking system through an API, with no rip-and-replace. Each transaction gets one of three outcomes: **Approve**, **Review** or **Block**. Every score comes with a plain-language reason and an audit record, so a human analyst always owns the final call on high-risk cases.

Problem. Legacy banks run 30–40-year-old mainframes that can’t be replaced quickly. Fraud teams still triage alerts by hand from static rules. Rules miss new patterns, flood analysts with false alarms, and can’t explain their decisions the way regulators (OCC, CFPB, EU AI Act) increasingly require.

Solution. An XGBoost scoring service sits beside the core system. It turns raw transactions into 12 behavioral signals, scores them, explains each score with SHAP, and routes the result through risk tiers. A logistic-regression challenger model keeps an auditable linear baseline running alongside it.

Results (proof of concept, 2,512 transactions, Kaggle 2024):

Metric	Result	What it means
ROC-AUC (test set)	1.000	Fraud and legitimate transactions fully separated
Recall	59 of 59 fraud cases caught	Zero missed fraud on the held-out set
Precision	0 false alarms in 444 legitimate transactions	Zero customer friction on the test set
10-fold stratified CV	ROC-AUC 1.000 ± 0.000 ; Recall 0.997 ± 0.010	The result isn’t a lucky split
Challenger (logistic regression)	ROC-AUC 0.9999; Recall 0.966	An explainable baseline is nearly as strong

Metric	Result	What it means
Inference	< 1 ms per transaction on CPU	Well within the 50 ms banking SLA, no GPU

Caveat up front. The dataset has no ground-truth fraud labels, so we engineered the label from domain rules. Perfect scores mostly show that the model recovers those rules. The real test is shadow mode on live labeled data (see Roadmap). That trade-off is a core PM learning from the project.

My role. Co-built with Aishwarya Somwanshi for EM 624 (Data Exploration & Informatics), Stevens Institute of Technology, Spring 2026. Scope covered problem framing, label and feature design, model evaluation, AI governance and a legacy-integration rollout plan. (Open: confirm which parts you owned versus Aishwarya before publishing.)

Problem, context and users

Fraud teams at legacy banks face four pressures at once, and today's rule engines handle none of them well.

- 1. **Losses keep growing.** Global fraud losses passed \$485B in 2023, and attackers now use ML to probe and bypass static rules.
- 2. **Accuracy is the wrong scoreboard.** Fraud is a small share of transactions (11.8% in our data, typically 1–5% in production). A model that approves everything looks 88.2% accurate here and catches zero fraud.
- 3. **Legacy cores can't be replaced.** Mainframe core banking systems are 30–40 years old. AI has to sit beside them as an overlay, not replace them, and decide within 50 ms per transaction.
- 4. **Regulators want reasons.** OCC Bulletin 2011-12 (model risk management), CFPB adverse-action rules and the EU AI Act (Articles 13 and 14) require explainable, auditable decisions with a human in the loop.

Stakeholders

Stakeholder	What they need	How FraudShield serves them
Fraud / risk analysts	A short, ranked queue with reasons, not a flood of alerts	Risk tiers, SHAP reason codes, one-click decisions

Stakeholder	What they need	How FraudShield serves them
Risk & compliance leads	Policy control, AML reporting, audit evidence	Tunable thresholds, 7-year write-once audit log, model cards
IT / legacy system owners	No core changes, predictable latency	REST scoring API, < 50 ms P99, rule-engine fallback
Product & platform teams	A scoring service with an SLA	Containerised, versioned model, blue-green deploys
Bank customers	Few false declines, fast resolution	Auto-approve for low risk, human review before any block
Regulators / auditors	Explainable, fair, governed models	Challenger model, fairness audits, decision lineage

Primary persona (illustrative)

Priya, Tier-2 fraud analyst at a regional bank. She clears about 300 alerts a shift from a rules engine, and most are false alarms. She can't tell a customer why a card was declined, because the rule only says "amount > threshold". She needs a prioritised queue, the top three reasons behind each flag, and confidence that low-risk cases are already handled.

Job to be done: "When a suspicious transaction comes in, help me decide in seconds whether to approve, hold or block, and give me a reason I can defend to the customer and the auditor."

Secondary persona (illustrative)

Marcus, Head of Fraud Risk. He sets the bank's risk appetite and answers to the regulator. He needs to move the approve/review/block thresholds, see the capture-rate vs. false-alarm trade-off before he commits, and produce model-governance evidence on demand.

Goals, success metrics and non-goals

The goal is to catch more fraud with fewer false alarms, inside the latency and governance limits of a legacy bank. Recall is the north-star metric, because a missed fraud is a direct financial loss.

Metric	Type	Target	PoC result
Fraud recall (capture rate)	North star	$\geq 95\%$	100% test; 99.7% $\pm$ 1.0% in 10-fold CV
ROC-AUC	Model quality	≥ 0.95 to deploy	1.000
Precision (1 – false-alarm share)	Guardrail	$\geq 90\%$	100%
Scoring latency	Guardrail	< 50 ms P99	< 1 ms model inference
Analyst review queue	Efficiency	Only the 0.30–0.80 band goes to humans	Queue sized by the threshold slider
False-positive-rate gap across segments	Fairness	< 20% between groups	Not yet measured (needs live data)
Decisions with a logged explanation	Governance	100%	Designed in (audit schema)
Manual review cost	Business	– 40–60%	Projection from the report, not measured

Non-goals for v1

- Replacing the core banking system or the existing rule engine. The rules stay as a fallback.
- Fully automated blocking with no human review. Every high-risk block goes to a senior analyst.
- Network or mule-ring detection across accounts, and sequence models such as LSTMs. Both need account-level history this dataset doesn't have.
- Customer-facing dispute UX. v1 gives analysts the reason codes they need to answer customers.

Data, insights and feature engineering

The [Kaggle Bank Transaction Dataset for Fraud Detection](#) (2024) has 2,512 synthetic transactions with 16 columns and no missing values. It covers April 2023 to November 2024 across ATM, Online and Branch channels and 40+ U.S. cities.

Labeling the data

The dataset has no fraud label, so we engineered one from banking-fraud domain rules:

```
isFraud = 1  if  TransactionAmount > 90th percentile ($701.31)
              OR (LoginAttempts > 3 AND < 30 min since the previous
transaction)
```

This flags 296 of 2,512 transactions (11.8%). As a robustness check, the written report also tested a stricter rule: at least 2 of {LoginAttempts $\geq$ 3, amount $> 2\times$ mean, amount $> 50\%$ of balance}. That gave a 5.0% fraud rate, and an optimised Random Forest still reached ROC-AUC 0.9997.

What the data says

Insight	Evidence	Product implication
Fraud is high-value	Fraud averages \$858.83 vs. \$222.63 legitimate (3.86 $\times$)	Amount relative to the population is the first signal to surface
Login behavior matters	LoginAttempts has the second-highest correlation with fraud ($ r = 0.299$)	Pair credential signals with amount; show both in reason codes
Online is riskiest	Online fraud rate 12.5% vs. ATM 11.8% and Branch 11.2%	Prioritise online in monitoring and in the analyst queue
Repeated logins are a strong tell	4+ logins: 69–88% fraud vs. 10% with one login (64 transactions). All timestamps fall 16:00–18:59, so time of day carries no real signal here	Surface login count in reason codes; collect full-day timestamps before using TxnHour
Accuracy misleads	All-legitimate baseline = 88.2% accuracy, 0% recall	Report recall and ROC-AUC, never accuracy alone

Features (12 model inputs)

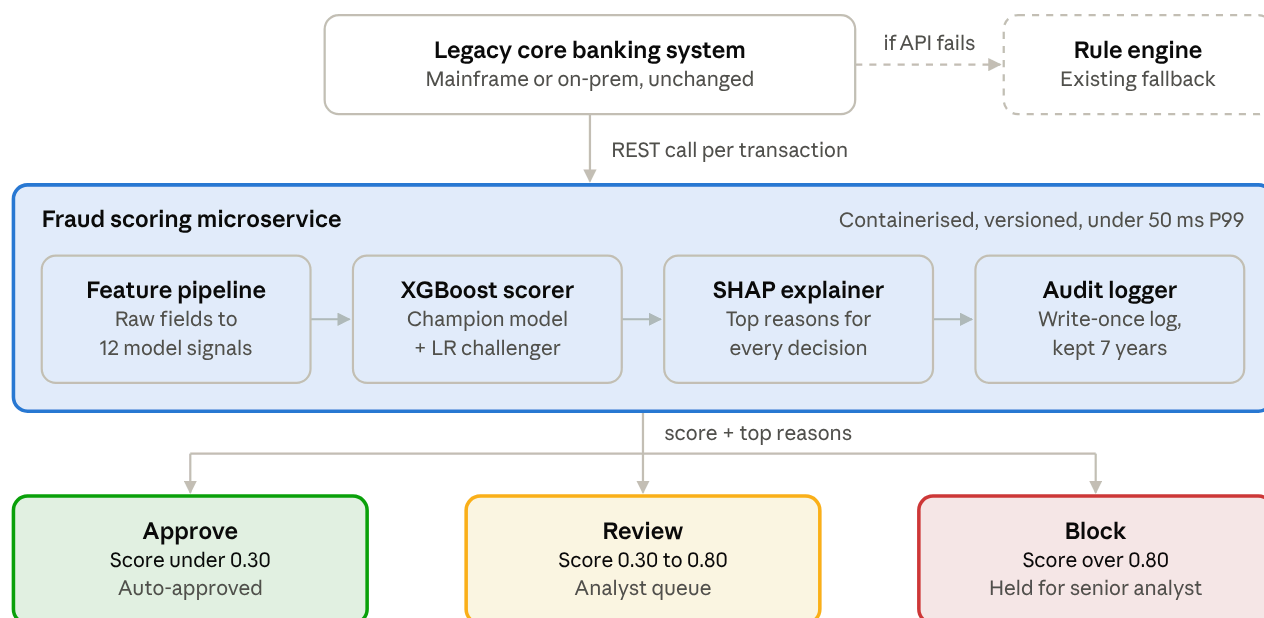
Feature	Kind	Fraud signal
TransactionAmount	Raw	Size of the transaction ($ r = 0.703$ with fraud)
AccountBalance	Raw	Account context for the amount
CustomerAge	Raw	Customer profile
LoginAttempts	Raw	Credential stuffing / brute force
TimeSincePrevTxn	Engineered	Rapid consecutive transactions suggest account takeover (median gap 0.17 h)
TxnHour	Engineered	Time-of-day pattern
HighLoginAttempts	Engineered flag	LoginAttempts > 1 (4.9% of transactions)
HighAmountFlag	Engineered flag	Amount in the top 10% (> \$701.31)
TransactionType_Enc	Encoded	Debit vs. credit
Ch_ATM, Ch_Branch, Ch_Online	One-hot	Channel

Identifiers (TransactionID, AccountID, DeviceID, IP, MerchantID) were dropped to prevent leakage. We encoded before scaling, fit scalers on the training data only, and used a stratified 80/20 split (2,009 train / 503 test, 11.8% fraud in each).

Solution: model, architecture and explainability

FraudShield is a decision-support layer, not a replacement for human judgment. The legacy core makes one REST call per transaction and gets back a score, a tier and the reasons behind it.

Scored beside the legacy core, routed to approve, review or block



FraudShield architecture · scoring service beside the legacy core

If the scoring API fails, the bank's existing rule engine takes over, so the service can never stop payments.

Model approach

1. **Baselines first.** We compared six classifiers from four families (Logistic Regression, Linear SVC, Decision Tree, Random Forest, Gradient Boosting, KNN) under both StandardScaler and MinMaxScaler.
2. **Champion: XGBoost.** 200 trees, learning rate 0.05, max depth 5, 80% row and column subsampling. `scale_pos_weight = 7.48` (legitimate-to-fraud ratio) corrects the class imbalance without resampling.
3. **Challenger: Logistic Regression.** It runs alongside as a fully linear, auditable baseline for regulators. If champion and challenger disagree, the case goes to review.
4. **Scale-up option: LightGBM.** Same results here, with 3–5× faster training expected above 100K rows.
5. **Validation.** We compared 10-fold stratified K-fold, leave-one-subject-out and leave-one-out. 10-fold stratified was chosen as the benchmark: lowest variance at only 10 model fits.

Explainability

Layer	Method	Who uses it
Global	XGBoost feature importance: HighAmountFlag 0.492, TransactionAmount 0.215, LoginAttempts 0.170, HighLoginAttempts 0.098	Model risk, auditors
Per decision	SHAP values: the top 3–4 signals that pushed this score up or down	Analysts, customer disputes
Baseline	Logistic-regression coefficients	Regulators wanting a linear benchmark

Decision policy (human in the loop)

Score	Tier	Action	Who decides
< 0.30	Approve	Transaction goes through	Automated
0.30–0.80	Review	Held in the analyst queue with reasons	Fraud analyst
> 0.80	Block	Held; customer contacted	Senior analyst confirms

No customer is permanently blocked without a human decision. The thresholds belong to the risk team, not the model.

Results

On the held-out test set (503 transactions, 59 fraud), the tree ensembles separated fraud perfectly. The linear challenger came within 0.0001 ROC-AUC of them.

Model comparison (StandardScaler, stratified 80/20 split)

Model	Accuracy	Precision	Recall	F1	ROC-AUC
XGBoost (champion)	1.000	1.000	1.000	1.000	1.000
LightGBM	1.000	1.000	1.000	1.000	1.000
Random Forest	1.000	1.000	1.000	1.000	1.000

Model	Accuracy	Precision	Recall	F1	ROC-AUC
Gradient Boosting	1.000	1.000	1.000	1.000	1.000
Decision Tree	1.000	1.000	1.000	1.000	1.000
Logistic Regression (challenger)	0.996	1.000	0.966	0.983	0.9999
Linear SVC	0.994	0.983	0.966	0.974	— (no probabilities)
KNN (k = 5)	0.988	0.965	0.932	0.948	0.998

MinMaxScaler hurt the scale-sensitive models (KNN recall fell to 0.864) and left the tree models unchanged, so StandardScaler is the production default.

XGBoost on the test set

	Predicted legitimate	Predicted fraud
Actually legitimate (444)	444 correct	0 false alarms
Actually fraud (59)	0 missed	59 caught

Is it stable? Cross-validation

Strategy	Accuracy	Recall	F1	ROC-AUC	Verdict
10-fold stratified	0.9996 ± 0.0012	0.997 ± 0.010	0.998 ± 0.005	1.000 ± 0.000	Recommended: low variance, 10 fits
Leave-one-out (300-row subset)	0.987 ± 0.115	—	0.107 ± 0.309	—	Too noisy for imbalanced data
Leave-one-subject-out	—	—	—	—	Not evaluated; see the note below

LOSO picked `TransactionID` as the subject column, so every hold-out held one row and was skipped. A fix is to group by `AccountID` (495 accounts), and it's on the v1.1 list. The written report's 3-rule label variant ran LOSO successfully (ROC-AUC 0.999 ± 0.002).

What the scores really tell us

The perfect scores are a finding, not a victory lap. The label is built from amount above the 90th percentile, and `HighAmountFlag` encodes that exact threshold. That's why `HighAmountFlag` carries 49% of XGBoost importance. The model is learning the labeling

rule. The useful product evidence is elsewhere:

- The pipeline, tiers, explanations and audit trail work end to end.
- A simple, explainable challenger nearly matches the champion, which makes governance easy.
- Signals outside the label rule still carry weight, such as TxnHour, CustomerAge and TimeSincePrevTxn in the report's Random Forest.
- The next real metric comes from shadow mode on confirmed fraud labels (see Roadmap).

Product requirements

v1 is an analyst console on top of the scoring API. Priority follows MoSCoW.

User stories

ID	As a...	I want to...	So that...	Priority
US-1	Fraud analyst	see a queue of transactions sorted by risk score	I work the riskiest cases first	Must
US-2	Fraud analyst	see the top reasons behind each score in plain language	I can decide fast and explain it to the customer	Must
US-3	Fraud analyst	approve, escalate or confirm fraud in one click, with a note	every decision is logged with my name	Must
US-4	Head of fraud risk	move the approve/review/block thresholds and preview the effect	I can set the bank's risk appetite with data	Must
US-5	Compliance officer	export an audit record for any decision	I can answer a regulator or a dispute	Must
US-6	Fraud analyst	see when the champion and challenger models disagree	borderline cases get a human look	Should
US-7	Head of fraud risk	see model health (drift, recall trend, FPR by segment)	I can catch decay and bias early	Should

ID	As a...	I want to...	So that...	Priority
US-8	Fraud analyst	score a hypothetical transaction ("what-if")	I understand how the model reacts	Could
US-9	Customer	get an instant, clear reason when a payment is held	I trust the bank and resolve it quickly	Won't (v1)

Functional requirements

ID	Requirement	Acceptance criteria
FR-1	Score API	<code>POST /score</code> accepts a transaction and returns <code>{score, tier, reasons[3], model_version}</code>
FR-2	Feature pipeline	Derives the 12 features (TxnHour, TimeSincePrevTxn, flags, encodings) with the scaler fitted at training time
FR-3	Risk tiering	Configurable thresholds, defaulting to 0.30 / 0.80; changes are versioned and need approver sign-off
FR-4	Explanations	SHAP top-3 contributions for every scored transaction, mapped to plain-language reason codes
FR-5	Challenger	LR challenger scores in parallel; a disagreement across a tier boundary routes the case to Review
FR-6	Analyst actions	Approve / Escalate / Confirm fraud, with a required note on overrides
FR-7	Audit log	Write-once record: transaction ID, timestamp, model version, score, decision, top features, reviewer
FR-8	Fallback	If the API errors or times out, the existing rule engine decides and the event is logged
FR-9	Monitoring	PSI on key features every 30 days; alert when recall drops more than 5%

Non-functional requirements

Area	Requirement
Latency	< 50 ms P99 end to end; model inference < 1 ms on CPU
Availability	99.99%, with active-passive failover to the rule engine
Deployment	Containerised, versioned models; blue-green deploys with zero downtime
Retention	Audit records kept 7 years (AML) in immutable storage
Security & privacy	Field-level encryption, least-privilege access to features; GLBA, PCI-DSS, GDPR
Fairness	False-positive-rate gap < 20% across age, channel and location segments, reviewed monthly
Explainability	100% of decisions carry a stored explanation ID

UX: core screens and flows

The console has five screens. The interactive demo and the Lovable prototype both build them.

Screen	Purpose	Key elements
1. Overview dashboard	Health of the fraud program at a glance	KPI tiles (transactions scored, flagged, recall, false alarms), tier split, fraud rate by channel and by hour
2. Alert queue	Analysts' daily work list	Table sorted by score; tier badges; filters by tier, channel and amount; challenger-disagreement flag
3. Transaction detail	One decision, fully explained	Score gauge, tier, transaction facts, top SHAP reasons as bars with plain-language text, Approve / Escalate / Confirm fraud, audit record preview

Screen	Purpose	Key elements
4. Threshold policy	Risk appetite tuning	Two sliders (review and block cut-offs), live recall, precision and queue size, with a change-approval note
5. Model & governance	Evidence for model risk and audit	Model card, comparison table, CV results, feature importance, fairness and drift checks, rollout phase

Analyst flow: from alert to decision

1. A transaction is scored; anything 0.30 or above lands in the **Alert queue**, riskiest first.
2. The analyst opens it and reads the score, tier and top three reasons, e.g. "Amount \$1,240 is in the top 10% of all transactions" and "4 login attempts before this payment".
3. They check context such as balance, channel, time since the last transaction and customer profile.
4. They choose **Approve**, **Escalate** to a senior analyst, or **Confirm fraud**. Any override of the model needs a note.
5. The decision, reviewer and explanation ID are written to the audit log. Confirmed outcomes become the real labels for the next retrain.

Risk-lead flow: changing the policy

1. Open **Threshold policy** and drag the review or block cut-off.
2. Watch recall, precision and daily review-queue size update live.
3. Submit the change with a rationale. It takes effect after a second approver signs off, and the change is versioned in the audit log.

Reason-code copy (analyst-facing)

Signal	Reason shown
HighAmountFlag / TransactionAmount	"Amount is in the top 10% of all transactions (over \$701)"
LoginAttempts / HighLoginAttempts	"Multiple login attempts before this transaction"
TimeSincePrevTxn	"Very soon after the previous transaction on the system"

Signal	Reason shown
TxnHour	"Unusual time of day for this activity"
AccountBalance	"Large relative to the account balance"

Risks, ethics and compliance

The biggest risk is shipping a model that only learned its own labeling rule. Every other risk below is real but more manageable.

Risk	Impact	Likelihood	Mitigation
Label determinism: the model mirrors the engineered rule	High: real recall will be lower than 100%	High	Shadow mode on confirmed fraud labels; time-based split; remove label-defining flags from features
Concept drift as fraudsters adapt	High	High	Monthly PSI on features; emergency retrain if recall drops > 5%; seasonal recalibration
False declines hurt customers	Medium: churn, complaints	Medium	Human review before any block; precision guardrail $\geq 90\%$; fast-path dispute handling
Bias by age, channel or location	High: regulatory and reputational	Medium	Monthly segment metrics; FPR gap < 20%; consider dropping Location; external audits (EU AI Act Art. 9)
Higher fraud rate in data (11.8%) than production (1-5%)	Medium: optimistic metrics	High	Re-test at realistic prevalence; precision-recall curves, not only ROC

Risk	Impact	Likelihood	Mitigation
Scoring outage stops payments	High	Low	Rule-engine fallback; 99.99% availability; blue-green deploys
Sensitive data exposure	High	Low	Data minimisation, field-level encryption, role-based access

Regulatory mapping

Requirement	How FraudShield meets it
OCC Bulletin 2011-12 / SR 11-7 (model risk management)	Model card, CV evidence, challenger model, versioning, monitoring
CFPB adverse-action notices	Per-decision SHAP reason codes in plain language
EU AI Act Art. 13 (transparency)	Logged explanation for every decision
EU AI Act Art. 14 (human oversight)	Review and Block tiers always need a human
AML record-keeping	7-year, write-once audit log
GLBA, PCI-DSS, GDPR	Encryption, access controls, minimal features

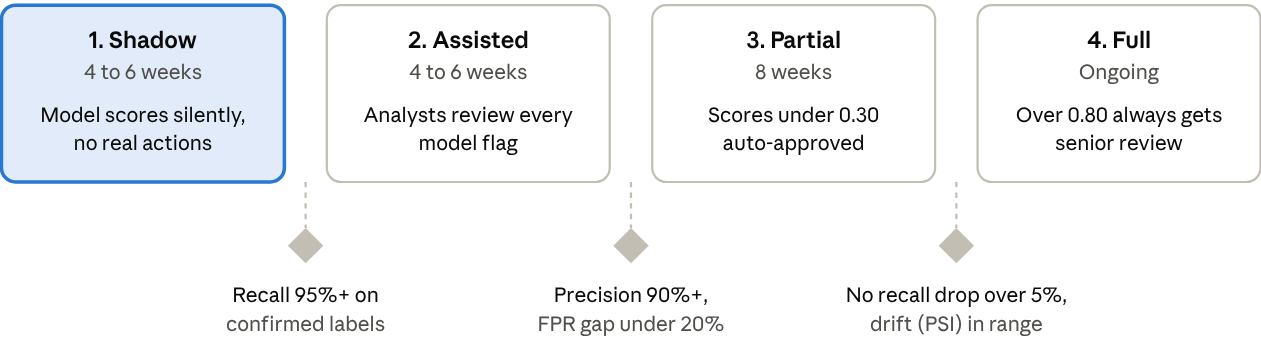
Limitations (stated openly)

- **Synthetic, engineered labels.** No confirmed fraud cases. This is a proof of concept for the product and pipeline, not proof of real-world accuracy.
- **Small dataset.** 2,512 rows is too small for sequence models (LSTMs, Transformers), and the data has no multi-transaction account histories.
- **No network view.** Mule networks, synthetic identities and social-engineering fraud are out of reach for single-transaction features.
- **LOSO not completed.** The grouping column needs fixing to measure performance on unseen customers.

Roadmap, go-to-market and learnings

Rollout follows the project's four-phase plan. Each step adds automation only after the model has proved itself on live, confirmed outcomes.

Automation widens only after each gate is met



Phased rollout · 4 phases, 3 proposed gates

Phases and durations come from the project. The gate criteria are proposed, using the success-metric targets above.

Product roadmap

Horizon	Item	Why
Now (v1.1)	Group LOSO by AccountID; time-based train/test split	Measure performance on unseen customers and future periods
Now (v1.1)	Retrain without HighAmountFlag; re-test at 1–3% fraud prevalence	Prove the model learns more than the labeling rule
Next (v2)	Real-time SHAP reason codes in the API response	Per-decision explanations for CFPB and EU AI Act
Next (v2)	Drift monitoring and automated retraining pipeline	Fraud patterns change constantly
Next (v2)	Bias audit: FPR by age group, channel and geography	Fairness guardrail before automation expands
Later (v3)	Partner bank data with confirmed fraud labels	Replace engineered labels with ground truth

Horizon	Item	Why
Later (v3)	Unsupervised anomaly detection (Isolation Forest, autoencoders)	Catch novel fraud absent from training data
Later (v3)	Sequence models (LSTM) on 50K–100K account histories	Detect multi-step account takeover

Go-to-market (concept)

- **Target customer:** regional and community banks and credit unions on legacy cores that can't afford a core replacement.
- **Wedge:** "Explainable fraud scoring in 50 ms, no core changes." Start in shadow mode, so there's no risk to the buyer during evaluation.
- **Buyers and users:** Head of Fraud / CRO buys; fraud analysts use it daily; compliance and model risk sign off.
- **Proof points to sell on:** recall uplift and false-alarm reduction measured in the bank's own shadow-mode data, plus a ready model-governance pack.
- **Pricing idea:** platform fee plus per-1,000-transactions scoring, with a paid pilot that converts after Phase 2.

Business case (illustrative)

The report modelled 100,000 transactions a day at 5% fraud prevalence and an \$860 average fraud value. On those assumptions, a model at 100% recall would stop about \$4.3M of fraud a day, and one at 88% recall about \$3.8M. The report also projects a 40–60% cut in manual review cost. These are scenario numbers, not measured results; real figures come out of shadow mode.

What I learned as a PM

1. **A perfect score is a question, not an answer.** 100% across five models pointed to label leakage. Spotting that early is a product skill, not only a data-science one.
2. **Pick the metric that maps to money.** Recall (missed fraud = lost money) leads; precision (false alarms = friction) is the guardrail. Accuracy alone would have reported 88% for a useless model.
3. **In regulated AI, governance is the product.** Thresholds owned by the risk team, reason codes, a challenger model and an audit log turn a model into something a bank can actually deploy.

4. **Integrate, don't replace.** An API overlay with a rule-engine fallback is what makes AI adoptable on a 40-year-old core.
5. **De-risk with phases.** Shadow → Assisted → Partial → Full lets the business trust the model one gate at a time.

Sources

- Project notebook, deck and report: *AI-Driven Fraud Detection in Legacy Banking Systems*, H. Borole & A. Somwanshi, EM 624, Stevens Institute of Technology, Spring 2026 (instructor: Dr. Sang Won Bae).
- Dataset: [Bank Transaction Dataset for Fraud Detection](#), Kaggle, 2024.
- Results re-run from the notebook code on the attached CSV; every metric above reproduces.