Recall-Oriented Fraud Detection on Synthetic Transactions

Nello Di Candia

November 3, 2025

Abstract

We present a practical, recall-oriented workflow for binary fraud detection on a structured, synthetic transaction dataset ($\sim 10\,000$ rows, ~2% fraud). We build preprocessing and modeling pipelines, optimize decision thresholds with respect to F_2 (favoring recall), and study the effect of threshold changes on false positives (FP), true positives (TP), and total signals. We compare a Logistic Regression baseline with SMOTE to tree ensembles (RandomForest and BalancedRandomForest) tuned with F_2. We document retuning the decision threshold to deliver ~10% more FPs—mimicking an operational push for more alerts—and analyze FP vs. signal curves and threshold–metric trade-offs. Artifacts (models, metrics, plots) are saved for reproducibility.

1. Introduction

Fraud detection often prioritizes recall: missing fraud (false negatives, FN) is costly, and teams will often accept increased false positives (FP) for better coverage. This paper documents an end-to-end applied workflow: synthetic data generation, baseline modeling, recall-focused training with SMOTE and F_2 scoring, threshold retuning to increase alerts, and threshold analysis to quantify trade-offs. The complete project is available at https://github.com/o35048y/fraud_detection_project.

2. Data

Source: `data/synthetic_transactions.csv` generated by `src/data_generator.py`. Size: ~10,000 transactions, fraud ratio $\approx 2\%$.

Features: Categorical: `country`, `channel`, `device_type`, `merchant_category`, `currency`. Numeric: `amount`, `hour`, `is_high_risk_country`, `is_international`, `card_present`, `velocity_24h`. Dropped IDs: `user_id`, `merchant_id`. Target: `label` (1 = fraud, 0 = non-fraud).

The generator introduces plausible patterns: log-normal amounts, time-of-day effects, high-risk and international flags, velocity features, and categorical context.

3. Methods

3.1. Preprocessing

One-hot encoding for categorical features (unknown-safe) and scaling for numeric features. Train/test split: 80/20, stratified by target with `seed = 42`.

3.2. Models

Logistic Regression (LR) with imbalanced-learn `Pipeline`: Preprocessor + SMOTE oversampling (tuned sampling ratio), `GridSearchCV` scored by F_2.

Tree ensembles: `RandomForestClassifier` (with/without `class_weight=balanced`) and `BalancedRandomForestClassifier` with joint grid search scored by F_2.

3.3. Threshold selection

The model outputs a probability score $p \in [0,1]$. A threshold τ converts scores into decisions: predict fraud if $p \geq \tau$, else non-fraud. Lower τ increases alerts (higher recall, lower precision); higher τ reduces alerts (lower recall, higher precision). We choose thresholds by maximizing F_2 on the test split and also retune τ to achieve ~10% more false positives.

3.4. Metrics

We report AUC-ROC, AUC-PR, confusion matrix (TN, FP; FN, TP), precision, recall, F_1, and F_2. The F_β measure is defined as

$$F_\beta = (1+\beta^2)\,\frac{\text{precision} \cdot \text{recall}}{\beta^2\,\text{precision} + \text{recall}}, \tag{1}$$

with $\beta = 2$ to prioritize recall.

4. Experiments and Artifacts

Scripts reside in `src/models/` and `scripts/`. Outputs are under `models/` and `analysis/`.

4.1. Logistic Regression (SMOTE + F_2)

Metrics at the F_2-optimal threshold ($\tau = 0.635$), saved in `models/metrics.json`: ROC AUC 0.9443; PR AUC 0.5786; F_1 0.5306; F_2 0.5963; confusion matrix $\big[[1928, 32], [14, 26]\big]$ (TN, FP; FN, TP). Retuning to `+10% FP`: τ 0.635 → 0.6180; FP 32 → 36; signals 58 → 62; FP/signals 0.552 → 0.581.

4.2. Tree Ensembles (RF & BalancedRF, F_2)

Best configuration (`models/tree_metrics.json`): `RandomForestClassifier`, `class_weight=balanced`, $n_{\text{estimators}} = 600$, max_depth = 12, min_samples_leaf = 3, max_features = sqrt. Metrics at F_2-optimal τ (0.3425): ROC AUC 0.9458; PR AUC 0.4185; F_1 0.3575; F_2 0.5351; confusion matrix $\big[[1853, 107], [8, 32]\big]$. Retuning `+10% FP`: τ 0.3425 → 0.3327; FP 107 → 118; signals 139 → 150; FP/signals 0.770 → 0.787.

5. Results

Table 1: Methods comparison: performance and efficiency

Method	F2	PR AUC	Model Size (MB)	Throughput (ex/s)
LogReg+SMOTE (F2)	0.5963	0.5786	0.0596	501,822
RandomForest (F2)	0.5351	0.4185	10.9378	19,316

Lowering τ increases both TP and FP; recall improves, precision drops. The LR model offers higher PR AUC and better precision at comparable recall levels than the tree model in this setup, while the tree model achieves strong recall at a much lower threshold with substantially more FPs. Retuning `+10% FP` shifts operating points along each curve as expected: more signals overall with modest recall gains.

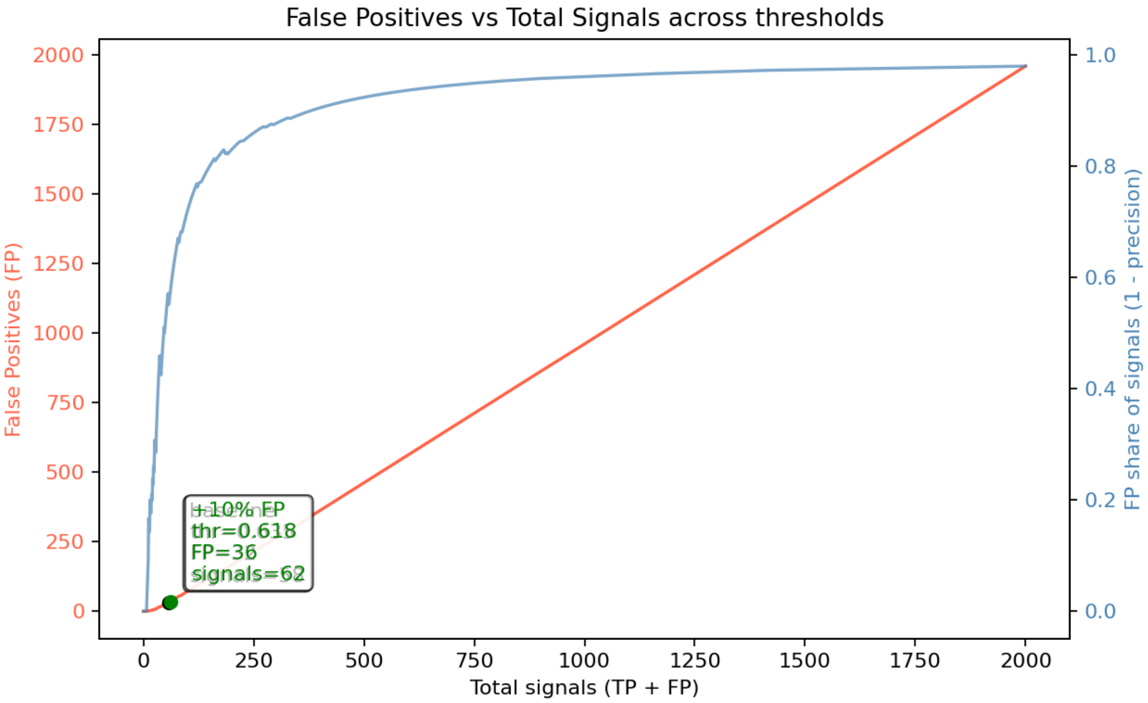

Figure 1: Logistic Regression: FP vs. total signals across thresholds; annotated baseline/retuned points.

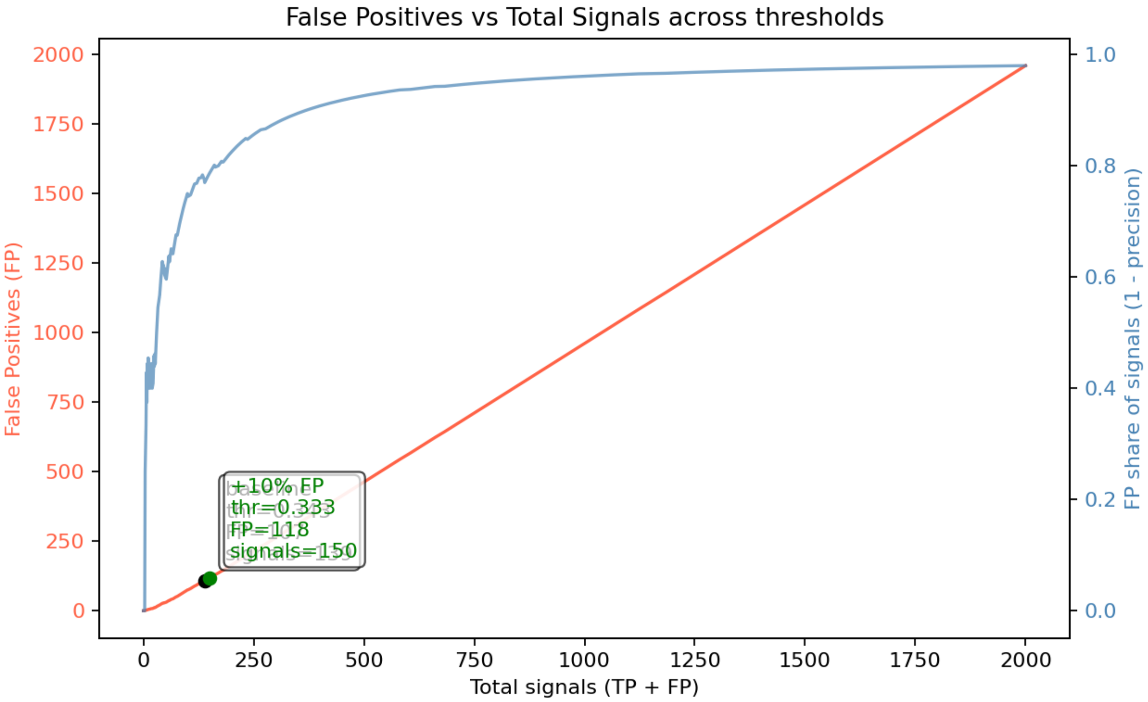

Figure 2: Tree Ensembles: FP vs. total signals across thresholds; annotated baseline/retuned points.

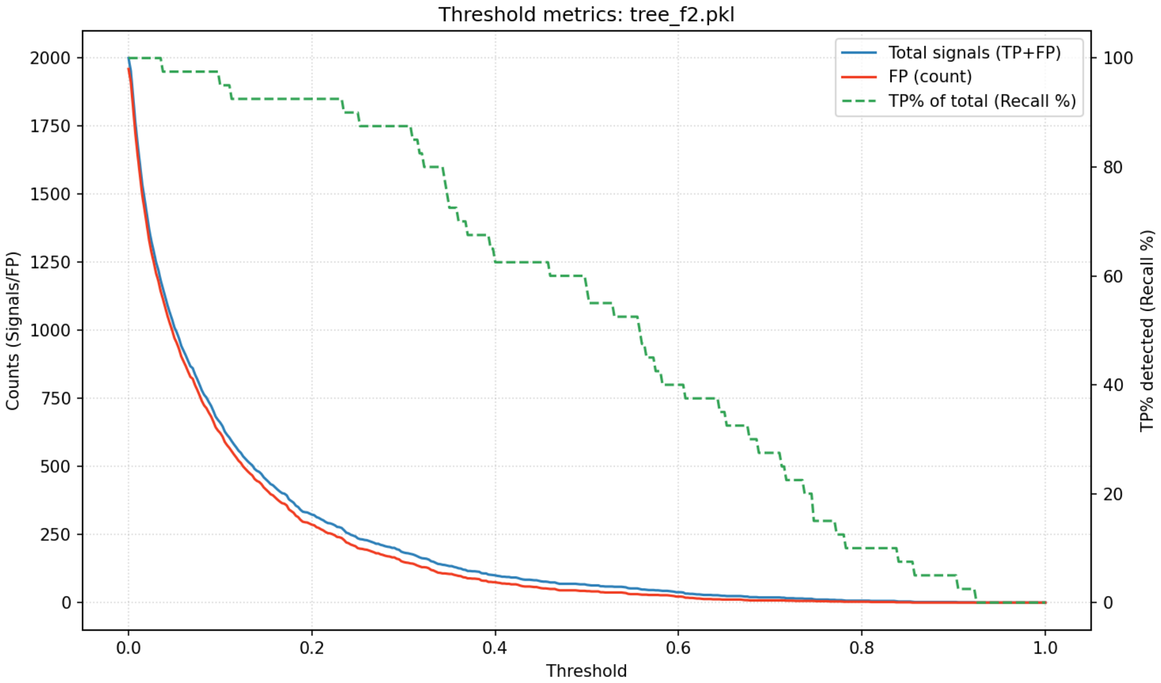

Figure 3: Tree Ensembles: Threshold sweep showing total signals (TP+FP), FP, and TP% detected (recall%).

6. Discussion

Threshold as a control: τ is an interpretable knob for operations. Selecting τ by F_2 favors catching more fraud; alternatively, set τ to match review capacity or a target FP rate.

Model comparison: On this dataset, LR+SMOTE reached better PR AUC, while RF delivered higher raw recall at the expense of precision. Choice depends on alert budget and tolerance for FPs.

Cost-sensitive view: If a false negative costs C_{FN} and a false positive costs C_{FP}, threshold selection can minimize expected cost by weighting these terms, i.e., optimize a cost-weighted utility.

7. Limitations

Synthetic data may not capture real-world biases, leakage, interaction effects, or distribution shift. Tree ensemble probabilities may require calibration (Platt/Isotonic) for stable thresholding. Hyperparameter grids were modest for speed; broader searches or Bayesian optimization may further improve results.

8. Future Work

Boosted trees (LightGBM/XGBoost) with F_2 optimization and probability calibration; segment-specific thresholds (e.g., by `channel`, `country`); drift monitoring and periodic threshold recalibration; cost-sensitive training/decisioning with explicit business cost ratios.

9. Reproducibility

Artifacts and scripts are provided in the repository. Key outputs reside in `models/` and `analysis/`. Environment: Windows, PowerShell; Python 3; core libs: NumPy, pandas, scikit-learn, imbalanced-learn, matplotlib. A script `scripts/verify_install.py` validates installed versions.