L03: Data-Driven Finance – Models, Governance, and Failure Modes

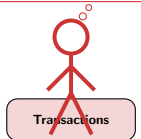
Extended Slides – BSc Digital Finance Course

Digital Finance

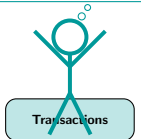
BSc Digital Finance Course

© Joerg Osterrieder 2026

"I have 5 years of data – why am I still wrong?"



"Different data, same problem, better answer."



More rows of the same data does not help. More kinds of data, well-governed, often does.

What Will You Be Able to Do After These Extended Slides?

1. Classify data sources in finance along the structured–semi-structured–unstructured spectrum and estimate signal decay for each
2. Select a model family (linear, tree-based, neural, unsupervised) for a given finance problem, arguing from data shape and regulator expectations
3. Write and read short Python snippets that train, evaluate, and interrogate a credit or fraud model
4. Apply cross-entropy, ROC/AUC, and permutation importance correctly, including their known failure modes
5. Audit a model for disparate impact, Shapley explainability, and Goodhart-style incentive distortion
6. Specify the five stages of the SR 11-7 / ECB TRIM model lifecycle and name what evidence each stage demands

---

Six objectives in five sections. 32 frames, 5 Python listings, 5 math frames, 5 charts. Pre-flight reading: basic probability, logistic regression, Python control flow, one prior exposure to cross-validation.

Outline

Which Three Data Shapes Cover Everything a Bank Ingests?

The shape spectrum.

- **Structured** – rows × columns, fixed schema. Transactions, positions, balances, corporate reference data.
- **Semi-structured** – tagged but not tabular. JSON, XML, SWIFT messages, SEC XBRL filings, Form 10-K tables.
- **Unstructured** – no schema. Earnings-call audio, news headlines, satellite imagery, voice-of-the-customer, PDFs scanned from 1998.

Why the split matters:

- Storage cost per byte rises right-to-left; value per byte usually rises left-to-right.
- Structured data is 20 percent of the bytes and 80 percent of the current revenue – but the opposite is true for marginal alpha.
- Regulators demand reproducibility; unstructured pipelines make reproducibility expensive.

Finance-specific examples:

Shape	Example	Typical size
Structured	Card authorisations	10 TB/yr
Structured	Market tick data	40 TB/yr
Semi-struct.	SWIFT MT/MX messages	1 TB/yr
Semi-struct.	10-K XBRL filings	60 GB/yr
Unstructured	Earnings-call audio	200 TB/yr
Unstructured	Satellite imagery	4 PB/yr

Signal decay. Tick data decays in milliseconds, credit features in months, satellite-derived parking counts in days. Decay rate should drive refresh budget, not storage budget.

The three shapes are not just a taxonomy – they predict cost, latency, and compliance effort. Decide storage and refresh budgets by signal decay, not by data volume.

What Exactly Is "Alternative Data" and Who Sells It?

Definition. Any dataset not produced by the financial system itself but predictive of a financial outcome.

Seven canonical categories:

- **Geospatial** – satellite imagery of parking lots, ports, oil-storage tank shadows
- **Web-scraped** – e-commerce prices, job postings, app reviews
- **Credit-card panels** – anonymised transaction panels (sold by Yodlee, Envestnet)
- **Consumer sentiment** – social-media scraping, Google Trends
- **Supply-chain** – shipping manifests, customs data
- **ESG telemetry** – emissions from CDP, satellite-derived CO_2 plumes
- **Device and app telemetry** – typing cadence, screen dwell, accelerometer

Market structure.

- Roughly 500 active vendors worldwide; 2024 spend estimated at USD 4 bn, growing around 25 percent year on year (Eagle Alpha, Neudata industry reports).
- Hedge funds, insurers, and retail-bank risk desks are the primary buyers.
- Median fund spend: USD 1–3 m per year across 8–15 datasets.

Four buyer questions before signing:

1. **Provenance** – where did the vendor get it, can the vendor prove consent?
2. **Exclusivity** – is the signal already priced in because 20 funds also bought it?
3. **Stability** – does last year's backtest reflect this year's panel?
4. **Legal** – GDPR, CCPA, web-scraping terms of service, material non-public information risk.

The alpha half-life of a newly published alternative dataset is typically 6–18 months before crowding erodes it.

Alternative data is a USD 4 bn market with around 500 vendors. The key buyer questions are provenance, exclusivity, stability, and legal exposure – technical quality comes fifth.

(c) Joerg Osterrieder 2026

Which Six Dimensions of Data Quality Should You Actually Measure?

Adapted from DAMA-DMBOK and Basel BCBS 239:

- **Completeness** – fraction of required fields present. Target above 99 percent for primary keys; graceful degradation elsewhere.
- **Accuracy** – agreement with a gold source or reconciled ledger. Measured by sampling plus back-testing against settled outcomes.
- **Timeliness** – arrival lag versus stated SLA. Time-decayed: a five-minute lag on a tick feed is fatal, a five-day lag on KYC data is routine.
- **Consistency** – same fact in two systems matches. Reconciliation breaks are the single biggest audit finding.
- **Uniqueness** – no duplicate entities. Often broken by merger integration and by fuzzy customer identifiers.
- **Validity** – conforms to defined format and business rules (for example currency codes in ISO 4217).

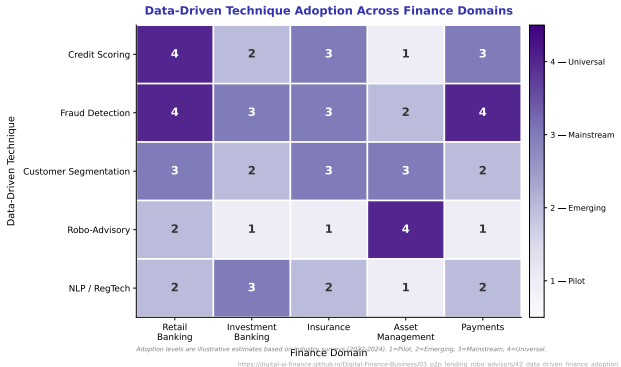
A minimal quality scorecard. Assign each source a 0–100 score per dimension. Weight by downstream use: a model-training source needs accuracy 95 and consistency 98, whereas a marketing dashboard can tolerate 85 and 90. Publish the scorecard; models inherit source scores and fail closed when scores drop.

Basel BCBS 239 is the binding standard for global systemically important banks (G-SIBs) – it codifies these six dimensions as regulatory obligations for risk-data aggregation.

Six quality dimensions: completeness, accuracy, timeliness, consistency, uniqueness, validity. Weight by downstream use. BCBS 239 is binding for global systemically important banks.

(c) Joerg Osterrieder 2026

How Fast Are Data-Driven Methods Actually Being Adopted in Finance?



What the adoption curve shows:

- **Credit scoring** is saturated – over 90 percent of consumer-credit decisions in the US and EU are machine-assisted.
- **Fraud detection** is saturating – most card networks have used gradient-boosted ensembles since roughly 2015.
- **AML/KYC screening** is mid-curve – adoption lags because false positives drive investigator cost, not direct revenue.
- **Anti-money-laundering rule tuning** is mid-curve – constrained by regulator expectations of auditability.
- **Algorithmic trading** is bimodal – deep at the top of the industry, shallow at the tail.

The lag pattern. Adoption tracks three drivers: direct revenue lift, regulator comfort, and whether humans still read every output. Low-revenue plus high-audit cases (AML) adopt last.

Adoption is not uniform. Revenue-linked uses (credit, fraud) saturate first; audit-heavy uses (AML, capital) lag. The lag is a reasonable signal of regulator risk tolerance, not a signal of technical limits.

Why Do Leading Banks Treat Data as a Product Rather Than a Project?

Project mindset (legacy):

- Data extracted once per use case, discarded afterwards
- Quality owned by the last analyst who touched it
- SLA measured in weeks; reconciliation manual
- Lineage lives in tribal memory

Product mindset (modern):

- Each dataset has a named owner, an SLA, and a versioned schema
- Consumers are first-class – they open tickets, not emails
- Breaking changes follow a deprecation cycle, not a cutover
- Lineage is machine-tracked (OpenLineage, Marquez)

Five artefacts that distinguish a data product:

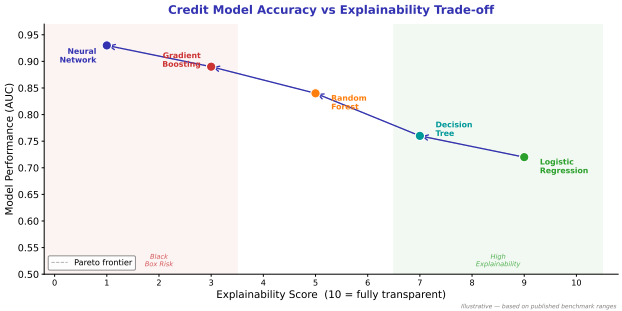
1. Contract (schema + semantics + SLA)
2. Discovery page in an internal catalog (for example Collibra, DataHub)
3. Observability (freshness, volume, distribution drift)
4. Access policy (data classification + row-level controls)
5. Retirement plan (sunset date or renewal criterion)

Why the shift. Regulators after the 2008 crisis mandated *risk-data aggregation* for major banks (Basel BCBS 239). Treating data as a product is the cheapest path to BCBS 239 compliance – project-style extracts cannot meet the aggregation standards.

The data-product mindset is not a fashion – it is the operating model that makes BCBS 239 and similar aggregation mandates feasible. Five artefacts: contract, catalog, observability, access policy, retirement plan.

(c) Joerg Osterrieder 2026

Which Model Family Fits Which Finance Problem – and Why?



Four dominant families in finance:

- **Linear / GAM** – logistic regression, generalised additive models. Transparent, regulator-friendly, brittle on interactions.
- **Tree ensembles** – XGBoost, LightGBM, CatBoost. The current workhorse for tabular credit and fraud.
- **Deep neural** – LSTMs for time series, transformers for text, CNNs for imagery. High capacity, expensive to validate.
- **Unsupervised** – PCA, isolation forest, DBSCAN. Used for anomaly detection and dimensional reduction.

Picking rule of thumb: start with logistic for credit; gradient boosting when you need 2–3 AUC points more; neural nets only when you have either unstructured inputs or more than 100,000 labelled training rows.

The accuracy-explainability trade-off is real but overstated. Tree ensembles give you 90 percent of neural accuracy with 10 percent of the validation cost – that is why they dominate production finance.

(c) Joerg Osterrieder 2026

What Does Cross-Entropy Actually Measure – and Why Is It the Right Loss?

Setup. Binary classifier predicts $\hat{p}_i \in (0,1)$ for observation i; truth is $y_i \in \{0,1\}$.
Binary cross-entropy (log loss).

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\Big]$$

Interpretation as Kullback-Leibler divergence. Cross-entropy minimisation is equivalent to minimising KL-divergence between the empirical label distribution and the predicted distribution, plus a constant (the entropy of the empirical distribution, which does not depend on the model parameters):

$$\mathcal{L}_{\mathrm{CE}} = H(y) + D_{\mathrm{KL}}(y \,\|\, \hat{p})$$

Why log and not squared error. Log loss penalises confident wrong predictions infinitely: as $\hat{p}_i \to 0$ for $y_i = 1$, the loss grows without bound. Squared error caps the penalty at 1, so it under-punishes overconfident mistakes.
Proper scoring. Log loss is *strictly proper*: its unique minimum is achieved when $\hat{p}_i$ equals the true conditional probability $\Pr(Y_i = 1 \mid X_i)$. This is why log loss – not accuracy – is the right training objective when probabilities will be used downstream (for expected-loss pricing, for example).

Cross-entropy is the KL-divergence between truth and prediction, up to a constant. It is strictly proper – its minimum coincides with the true conditional probability. That is why probability-consuming models train on log loss.

How Do You Train, Calibrate, and Score a Credit Logistic Regression in 22 Lines?

```python
# End-to-end logistic regression for binary credit-default classification
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import log_loss, roc_auc_score

# Features: income, debt_ratio, utilisation, age, credit_history_months
X = np.array([
    [52000, 0.22, 0.18, 34, 120],
    [31000, 0.48, 0.72, 22, 18],
    [78000, 0.15, 0.09, 41, 180],
    [24000, 0.55, 0.88, 27, 24],
    [65000, 0.28, 0.31, 38, 144],
])
y = np.array([0, 1, 0, 1, 0])  # 1 = default within 12 months

X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.4, random_state=7)
scaler = StandardScaler().fit(X_tr)
clf    = LogisticRegression(C=1.0, penalty="l2").fit(scaler.transform(X_tr), y_tr)
p_te   = clf.predict_proba(scaler.transform(X_te))[:, 1]
print(f"log loss = {log_loss(y_te, p_te):.3f}, AUC = {roc_auc_score(y_te, p_te):.3f}")
print("coefficients:", dict(zip(
    ["inc","dti","util","age","hist"], clf.coef_[0].round(3))))
```

This is the minimum viable credit scorer. Scaling is mandatory (otherwise income dominates); L2 penalty controls overfitting; probabilities (not classes) are scored so downstream expected-loss calculations stay honest.

Can You Beat Logistic Regression With Gradient Boosting in 20 Lines?

```python
# Gradient-boosted trees for tabular credit default
import numpy as np, xgboost as xgb
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_auc_score

# Assume X is (n, p) feature matrix and y is a binary label vector,
# both already prepared from a loan-application warehouse extract.
# Use stratified CV to preserve default-rate balance in each fold.
params = dict(
    objective   = "binary:logistic",
    eval_metric = "auc",
    max_depth   = 4,
    eta         = 0.05,
    subsample   = 0.8,
    colsample_bytree = 0.8,
    min_child_weight = 10,
    reg_lambda  = 1.0,
)

cv_auc = []
for tr_idx, te_idx in StratifiedKFold(n_splits=5, shuffle=True, random_state=3).split(X, y):
    dtr = xgb.DMatrix(X[tr_idx], y[tr_idx])
    dte = xgb.DMatrix(X[te_idx], y[te_idx])
    m   = xgb.train(params, dtr, num_boost_round=400,
                    evals=[(dte, "val")], early_stopping_rounds=30, verbose_eval=False)
    cv_auc.append(roc_auc_score(y[te_idx], m.predict(dte)))
print(f"mean CV AUC = {np.mean(cv_auc):.3f} +/- {np.std(cv_auc):.3f}")
```

Five-fold stratified CV with early stopping is the production default for tree ensembles. Expect around 2–4 AUC points over a well-tuned logistic on richer tabular datasets, at the cost of harder explainability.

Which Features Actually Matter – and How Do You Prove It?

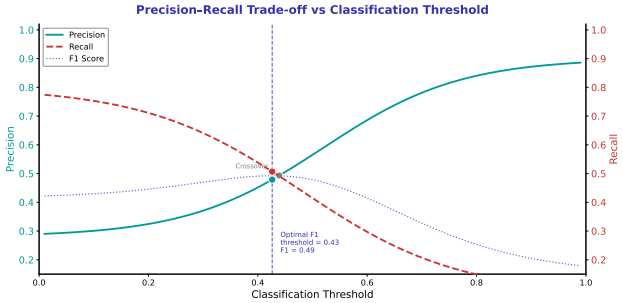
Permutation importance. For a trained model f, scored on holdout $\mathcal{D}_{\text{val}}$ with metric m (higher is better), define the importance of feature j:

$$\text{PI}_j = m\big(f, \mathcal{D}_{\text{val}}\big) - \mathbb{E}_\pi\Big[m\big(f, \mathcal{D}_{\text{val}}^{(j,\pi)}\big)\Big]$$

where $\mathcal{D}_{\text{val}}^{(j,\pi)}$ is the validation set with feature j permuted. Repeat with R random permutations to estimate the expectation.
Why permutation and not coefficient magnitude. Tree ensembles do not expose coefficients; and even for linear models, coefficient magnitude confounds scale with importance. Permutation importance is model-agnostic and scale-free.
Gotcha 1: correlated features. If features x_j and x_k are highly correlated and both carry the same signal, permuting j alone leaves the information accessible via k. Both will look unimportant individually.
Gotcha 2: extrapolation. Permutation creates feature combinations that never occur in the data-generating distribution (for example a 70-year-old with 2 years of credit history). The model is evaluated outside its training support – importance can be inflated.
Mitigation. Use *grouped* permutation (permute correlated blocks together) and *conditional* permutation (permute within strata of correlated variables).

Permutation importance is the default model-agnostic feature-importance measure. Two failure modes: correlated features are under-credited, and extrapolated feature combinations are over-weighted. Grouped or conditional permutation fixes both.

Precision, Recall, AUC: Which Metric Matches Which Business Question?



Metric-to-question mapping:

- **AUC-ROC** – discrimination across thresholds. Use when downstream will pick its own threshold.
- **AUC-PR** – precision-recall area. Use when the positive class is rare (under 5 percent) – fraud, default, sanctions hits.
- **Precision at k** – "of the top k we alert on, how many are true?" Use when investigator capacity caps k.
- **Recall at fixed false-positive rate** – "catching rate when we accept a 1 percent nuisance rate" – use in AML.
- **Expected calibration error (ECE)** – "do predicted probabilities actually occur?" Use when probabilities feed expected-loss pricing.

The trap. A 0.95 AUC model can still be unusable: it might have 40 percent precision at the operating threshold the business can staff. Always co-report rank metric (AUC) and an operating-point metric (precision at a business-feasible recall).

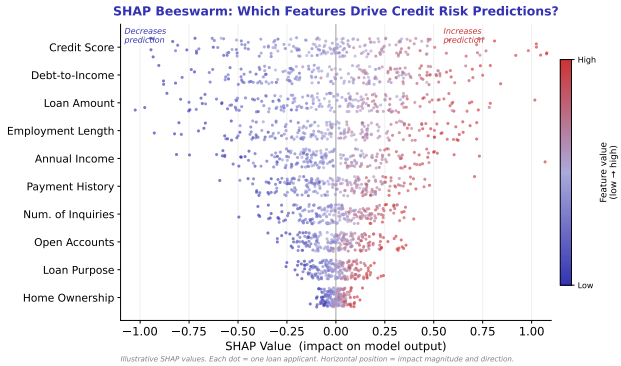
Rank metrics (AUC) and operating-point metrics (precision at k) answer different questions. Rare-positive problems also need AUC-PR. Always co-report with calibration (ECE) when probabilities will be used in pricing.

How Do You Convert an Earnings Call Into a Bullish-Bearish Signal?

```python
# FinBERT sentiment classifier for earnings-call transcripts
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

# FinBERT is a BERT model pre-trained on financial text; labels are
# 0 = negative, 1 = neutral, 2 = positive. We use it in inference mode only.
MODEL = "ProsusAI/finbert"
tok = AutoTokenizer.from_pretrained(MODEL)
mdl = AutoModelForSequenceClassification.from_pretrained(MODEL).eval()

def sentence_score(text: str) -> dict:
    """Return softmax probabilities over (negative, neutral, positive)."""
    ids  = tok(text, return_tensors="pt", truncation=True, max_length=256)
    with torch.no_grad():
        logits = mdl(**ids).logits
    probs = torch.softmax(logits, dim=-1)[0].tolist()
    return dict(neg=probs[0], neu=probs[1], pos=probs[2])

print(sentence_score(
    "We see continued headwinds in EMEA and have lowered fiscal-year guidance."))
# -> {'neg': 0.88, 'neu': 0.09, 'pos': 0.03}
```

FinBERT gives usable sentiment probabilities out of the box for sell-side research and news-scored trading. Production uses require topic segmentation, speaker diarisation, and drift monitoring on new analyst phrasings.

Can You See the Fingerprint of an Entire Model on a Single Chart?



Reading a SHAP beeswarm:

- Each dot is one observation.
- Horizontal position is the Shapley contribution to that prediction (positive = pushes toward default).
- Colour encodes the raw feature value (red high, blue low).
- Features are ordered top-down by mean absolute Shapley value.

Diagnostics at a glance:

- A clean red-right / blue-left pattern means the feature is monotonic – regulators prefer this.
- A red dot on the left means a high feature value can *lower* default probability for some observations – evidence of interactions.
- A bimodal horizontal spread means the feature matters only for a subgroup.

SHAP beeswarms are the single most informative explainability chart for tree ensembles.

A SHAP beeswarm compresses global and local explanations onto one chart: ordering gives global ranking; per-dot position gives local contribution. Monotone colour gradients indicate regulator-friendly features.

What Game-Theoretic Principle Guarantees a Fair Feature Attribution?

The attribution problem. Given prediction $f(x)$, split $f(x) - \mathbb{E}[f(X)]$ into per-feature contributions.
Shapley value. Let $N = \{1, \ldots, p\}$ be the feature set, $f_S(x)$ the model that uses only features in subset $S \subseteq N$ (typically via conditional expectation). The Shapley value of feature j is:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left[f_{S \cup \{j\}}(x) - f_S(x) \right]$$

Four fairness axioms (Shapley 1953, adapted to ML by Lundberg and Lee 2017):

1. **Efficiency**: $\sum_j \phi_j = f(x) - \mathbb{E}[f(X)]$ – contributions add up to the model's prediction offset.
2. **Symmetry**: if two features make the same marginal contribution in every coalition, they get equal credit.
3. **Dummy**: a feature that never changes the prediction gets zero.
4. **Additivity**: for an ensemble $f = f_1 + f_2$, $\phi_j(f) = \phi_j(f_1) + \phi_j(f_2)$.

Uniqueness. Shapley values are the *only* attribution satisfying all four axioms. This is why they are the regulator-preferred local explanation.
Computational cost. Exact computation is $O(2^p)$. For tree ensembles, TreeSHAP computes exact Shapley values in $O(TLD^2)$ where T = trees, L = leaves, D = depth – making it practical at scale.

Shapley values are the unique fair feature attribution. Efficiency, symmetry, dummy, additivity uniquely characterise them. TreeSHAP makes exact computation tractable for gradient-boosted models.

Which Four Definitions of Algorithmic Fairness Can You Actually Satisfy?

Let A be a protected attribute (for example ethnicity, gender), Y the true label, $\hat{Y}$ the prediction.

- **Demographic parity.** $\Pr(\hat{Y} = 1 \mid A = a) = \Pr(\hat{Y} = 1 \mid A = b)$. "Equal approval rates across groups."
- **Equal opportunity.** $\Pr(\hat{Y} = 1 \mid Y = 1, A = a) = \Pr(\hat{Y} = 1 \mid Y = 1, A = b)$. "Equal true-positive rates."
- **Equalised odds.** Equal TPR *and* equal FPR across groups.
- **Calibration within groups.** $\Pr(Y = 1 \mid \hat{p} = s, A = a) = s$ for every a. "A 20 percent predicted probability means 20 percent default rate in every group."

Impossibility result (Chouldechova 2017, Kleinberg et al. 2016). If base rates $\Pr(Y = 1 \mid A)$ differ across groups, no classifier can satisfy calibration and equalised odds simultaneously (except the trivial perfect or random classifier).

Regulatory mapping.

- US fair-lending (ECOA) tends toward **disparate impact** – a close relative of demographic parity, measured via the four-fifths rule (minority approval rate at least 80 percent of majority rate).
- EU AI Act emphasises process auditability more than a specific statistical definition.
- The COMPAS recidivism case (ProPublica 2016) centred on a violation of equal FPR between Black and white defendants, despite group-calibrated scores.

Four fairness definitions, one impossibility theorem: calibration and equalised odds cannot both hold when base rates differ. Regulators care about process audit, not a single statistical definition. Pick, document, and defend.

How Do You Check a Credit Model for Disparate Impact in 15 Lines?

```python
# Four-fifths rule disparate-impact audit (US ECOA / OCC 2010-16 guidance)
import numpy as np
from scipy.stats import fisher_exact

def disparate_impact(y_pred, protected, reference_group):
    """Return approval-rate ratio plus a Fisher-exact p-value vs reference."""
    approve_rate = {}
    for g in np.unique(protected):
        mask = protected == g
        approve_rate[g] = y_pred[mask].mean()

    ref = approve_rate[reference_group]
    for g, rate in approve_rate.items():
        ratio = rate / ref if ref > 0 else float("nan")
        a = int(((protected == g) & (y_pred == 1)).sum())
        b = int(((protected == g) & (y_pred == 0)).sum())
        c = int(((protected == reference_group) & (y_pred == 1)).sum())
        d = int(((protected == reference_group) & (y_pred == 0)).sum())
        _, p = fisher_exact([[a, b], [c, d]])
        flag = "FAIL" if ratio < 0.8 else "ok"
        print(f"group={g}: approve={rate:.3f}, ratio={ratio:.2f} [{flag}], p={p:.3g}")

# Example: 0 = approve, 1 = decline -> use 1 - decision as y_pred
# disparate_impact(1 - model.predict(X_val), protected, reference_group=0)
```

The four-fifths rule is a screening tool, not a verdict. A ratio below 0.80 obliges the lender to demonstrate business necessity (OCC 2010-16). Fisher-exact adds a significance check against small-sample noise.

What Do COMPAS, the Apple Card, and Amazon's Recruiter Teach Us About Hidden Bias?

COMPAS (2016).

- Northpointe's recidivism-risk score, used in US sentencing decisions.
- ProPublica showed Black defendants faced roughly double the false-positive rate of white defendants, despite within-group calibration.
- Surfaced the fairness impossibility theorem in public policy.

Apple Card (2019).

- Goldman Sachs issued Apple Card with spouse-differentiated credit limits – a 20× gap in some households.
- New York DFS investigated but cleared Goldman because no protected-class variable was directly used.
- Demonstrated that proxy variables (account history, income shape) can produce disparate outcomes even when the protected attribute is excluded.

Amazon recruiting (2018).

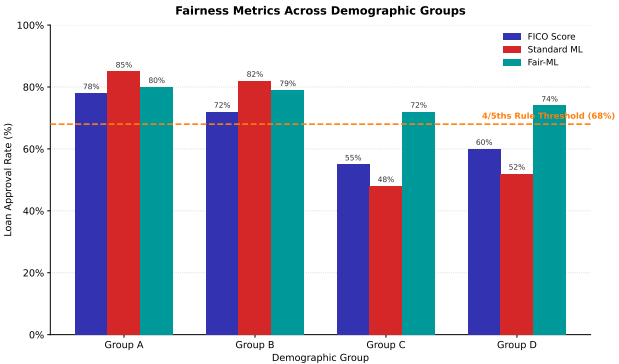
- Internal hiring-screen model penalised CVs with "women's" college names or the word "women's" in activities.
- The training set – Amazon's historical hires – was heavily male.
- Project was killed when auditors proved the bias could not be removed without removing the signal Amazon wanted to capture.

Three lessons:

1. Excluding protected attributes is necessary but far from sufficient.
2. Within-group calibration and cross-group FPR parity trade off – pick the one your domain cares about.
3. If the training history is biased, the model is biased. No algorithm rescues you from a biased label.

Three landmark cases, three lessons: proxy leakage defeats blinding; calibration does not imply equal errors; biased labels guarantee biased models. Auditors demand all three tests.

Which Fairness Metric Should You Report When You Cannot Satisfy Them All?



Reporting discipline.

- Pick a *primary* fairness metric that matches your legal exposure (disparate impact in US consumer credit; equal TPR in criminal-justice applications; calibration in insurance).
- Co-report at least one *secondary* metric to show you have looked at trade-offs.
- Publish group base rates alongside – otherwise a "fair" number can hide a population composition shift.

Regulator expectation. EU AI Act Annex III high-risk systems require a "risk management system" that documents the fairness metrics considered, the one chosen, and the rationale. The documentation itself – not any single number – is the compliance artefact.

Practical default. Report three metrics: *disparate impact ratio*, *equal-opportunity difference*, *calibration Brier decomposition*. This covers the three non-equivalent families.

You cannot pick a metric and forget the others. Report a primary plus a secondary, explain the trade-off, publish base rates. The EU AI Act judges the rationale, not the number.

(c) Joerg Osterrieder 2026

What Do Fed SR 11-7 and ECB TRIM Actually Require of a Model?

Fed SR 11-7 (2011) – Guidance on Model Risk Management. Applies to US bank holding companies above the thresholds set by the Fed (extended over time to mid-sized institutions).

- Defines a **model** broadly: any "quantitative method, system, or approach" that transforms inputs into outputs used in decisions.
- Mandates **three lines of defence**: model developer, independent validator, internal audit.
- Requires a firm-wide **model inventory** with risk-tiering.
- Each model needs documented *conceptual soundness*, *ongoing monitoring*, and *outcomes analysis*.

ECB TRIM (2017–2021) – Targeted Review of Internal Models. On-site review of internal risk models across 60+ significant European banks.

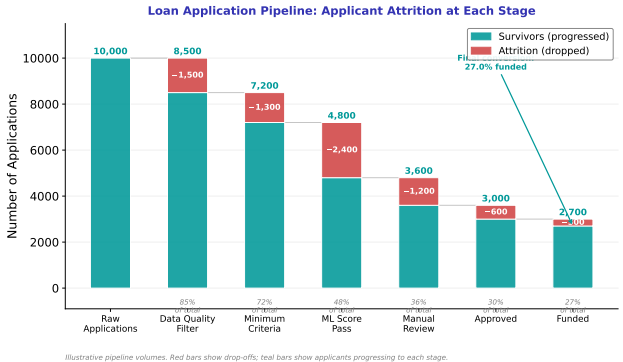
- Surfaced systematic weaknesses in data quality, model design choices, and validation independence.
- Drove specific capital add-ons where validation evidence was insufficient.
- Established the modern European model-governance bar.

UK PRA SS1/23 (2023) – Model Risk Management. Extends SR 11-7-style guidance to the UK with AI/ML-specific expectations.

Global convergence. SR 11-7, TRIM, SS1/23, and Basel BCBS 239 together have produced a de facto standard: inventory, tier, validate, monitor, retire, document.

Three regulations – SR 11-7, TRIM, SS1/23 – converge on the same six activities: inventory, tier, validate, monitor, retire, document. The detail differs by jurisdiction; the structure does not.

Where Does the Data Actually Live Between Arrival and Prediction?



Illustrative pipeline volumes. Red bars show drop-offs; teal bars show applicants progressing to each stage.

Eight pipeline stages:

1. **Ingest** – Kafka / Kinesis / SFTP drop
2. **Validate** – schema, range, completeness checks
3. **Enrich** – join reference data, compute features
4. **Store** – bronze / silver / gold layers in a data lake
5. **Train** – feature store snapshot to model
6. **Serve** – model registry to API endpoint
7. **Monitor** – feature and prediction drift, latency
8. **Govern** – lineage, access logs, audit trail

Where delays come from. Typically 60–80 percent of elapsed time is spent in stages 1–3 (ingest/validate/enrich), not in model training. Optimisation effort usually targets the wrong stage.

Reproducibility hinge. Every stage must stamp a version id; without per-stage versioning, "reproduce yesterday's decision" becomes unanswerable.

Eight stages from raw data to served prediction; most of the time lives in stages 1–3. Per-stage version stamping is what makes a decision reproducible six months later when the regulator asks.

What Evidence Does an Independent Validator Actually Demand?

Conceptual soundness.

- Documented theoretical basis; cited literature
- Justification for model family vs alternatives considered
- Documented feature engineering choices and exclusions
- Known limitations, with monitoring plan per limitation

Data.

- Lineage traceable to source systems
- Representativeness analysis vs deployment population
- Train/validation/holdout split documentation
- Treatment of missingness, outliers, class imbalance

Implementation.

- Code review sign-off
- Unit tests on feature transformations
- Reproducibility: same inputs produce same outputs

Performance.

- Primary metric plus two secondary metrics on holdout
- Stability analysis across time periods and segments
- Benchmark comparison (versus challenger and simple baseline)
- Sensitivity analysis: how much does output move when inputs shift?

Fairness / outcome.

- Disparate-impact analysis per protected class
- Adverse-action reason generation (ECOA / Regulation B)
- Ongoing outcomes vs predicted performance tracking

Operational.

- Fallback procedure when the model is unavailable
- Human override procedure and audit trail
- Retirement / re-training triggers defined in advance

A validator asks for evidence in five buckets – conceptual, data, implementation, performance, operational – plus a standalone fairness dossier. Missing evidence is what triggers capital add-ons, not poor accuracy.

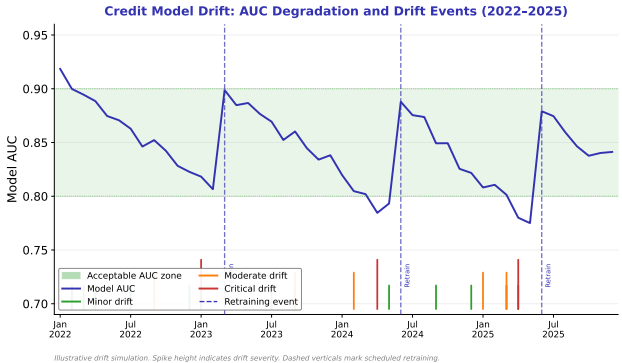
(c) Joerg Osterrieder 2026

How Do You Automate an Early-Warning Drift Alert in 18 Lines?

```python
# Population Stability Index (PSI) drift detection over binned feature
import numpy as np

def psi(expected, actual, n_bins=10, eps=1e-6):
    """PSI between two 1-D arrays. Convention: >0.25 is a material shift."""
    edges = np.quantile(expected, np.linspace(0, 1, n_bins + 1))
    edges[0], edges[-1] = -np.inf, np.inf
    e_hist, _ = np.histogram(expected, bins=edges)
    a_hist, _ = np.histogram(actual, bins=edges)
    e_pct = e_hist / e_hist.sum()
    a_pct = a_hist / a_hist.sum()
    mask = (e_pct > eps) & (a_pct > eps)
    return float(np.sum((a_pct[mask] - e_pct[mask]) *
                        np.log(a_pct[mask] / e_pct[mask])))

# Example: compare credit-utilisation feature between training window and
# the rolling 30-day production window. Alert when PSI crosses 0.25.
psi_util = psi(train_util, prod_util_last_30d)
if psi_util > 0.25:
    raise RuntimeError(f"drift: utilisation PSI={psi_util:.3f}")
elif psi_util > 0.10:
    print(f"watch: utilisation PSI={psi_util:.3f} (investigate)")
```

PSI under 0.10 is noise, 0.10–0.25 is a watch, over 0.25 is a material shift. Alert only on rolling windows to suppress day-of-week noise. PSI is the industry default for feature-level drift.

What Does the Full Anatomy of a Model Decay Look Like in Production?



Illustrative drift simulation. Spike height indicates drift severity. Dashed verticals mark scheduled retraining.

Three drift patterns:

- **Covariate shift** – $\Pr(X)$ changes; for example card-fraud geography shifts after a new merchant launch.
- **Label shift** – $\Pr(Y)$ changes; default rates rise in a recession.
- **Concept drift** – $\Pr(Y \mid X)$ changes; fraud tactics evolve to evade the last model.

Detection stack.

- Feature-level: PSI, Kolmogorov-Smirnov, Wasserstein
- Prediction-level: score distribution drift
- Performance-level: rolling AUC, lift decay
- Outcome-level: realised default rate vs predicted

Retraining triggers are written in advance: a calendar trigger (quarterly), a threshold trigger (PSI over 0.25), and an outcome trigger (realised default rate above predicted by more than 1 standard deviation).

Drift has three flavours, each needing its own detector. Retraining triggers should be codified in advance; ad-hoc "it felt wrong" retrains are audit red flags.

(c) Joerg Osterrieder 2026

Why Does a Metric Become Useless Once You Optimise for It?

Goodhart's Law (Goodhart 1975, Strathern 1997): "When a measure becomes a target, it ceases to be a good measure."

Four canonical finance examples:

1. **Credit-utilisation scores.** Lenders learned that "keep utilisation under 30 percent" improves FICO. Consumers learned the same rule. The signal now measures FICO-literacy, not credit worthiness.
2. **AML alert rates.** Investigator teams were measured on "alerts cleared per day". Rules were tuned to generate easy-to-clear alerts. Suspicious-activity-report (SAR) quality dropped.
3. **Quarterly earnings.** Management optimised quarterly accruals. Smoothing became a signal of manipulation, not stability.
4. **VaR-based risk limits.** Banks optimised book composition for low VaR. Tail risk migrated to the long-dated, illiquid positions VaR does not capture.

Why it happens formally. The population distribution responds to the model's decisions – *performative prediction* (Perdomo et al. 2020). The deployed model changes the data-generating process, invalidating the stationarity assumption that justified it.

Countermeasures:

- Use ensembles of metrics, not a single optimisation target.
- Hold back a "silent" detection model that is not itself used for decisions.
- Re-validate on a rolling holdout that post-dates the deployment, not the training data.

Goodhart's Law is the structural reason finance metrics decay. The fix is not a better metric – it is an ensemble of metrics and a silent-model backstop that is not itself gamed.

(c) Joerg Osterrieder 2026

Which Feedback Loops Quietly Corrupt Deployed Credit and Fraud Models?

Selection feedback.

- Only approved applicants generate repayment labels.
- "Would this rejected applicant have defaulted?" is unanswerable – the counterfactual is missing.
- Rejects get worse over generations because the model only learns from accepts.
- **Mitigation:** reject-inference techniques (parcelling, augmentation); periodic stochastic acceptances to probe the frontier.

Pricing feedback.

- High-risk customers quoted high rates; high rates cause defaults.
- Model "confirms" its own prediction.
- **Mitigation:** decouple risk score from price shaping; monitor pricing elasticity per segment.

Dataset-polluting feedback.

- Model outputs get logged into the very warehouse that trains the next model.
- "Predicted default" slowly becomes a feature of "actual default".
- **Mitigation:** separate inference and training storage; model-output fields flagged and excluded from training unless explicitly desired.

Attention feedback (fraud).

- Investigators probe where the model flags; unflagged segments stay unobserved.
- New fraud tactics develop in the unlit zone.
- **Mitigation:** reserve 2–5 percent of investigator capacity for randomly-selected cases ("canary sampling").

Common thread. Every feedback loop narrows the data-generating distribution the model sees. The fix is always some form of deliberate, budgeted, randomised exploration.

Four feedback loops: selection, pricing, dataset-polluting, attention. All narrow the model's view of reality. Countermeasure: budgeted randomised exploration – reject-inference probes, canary sampling, output-flag fencing.

(c) Joerg Osterrieder 2026

What Does an Adversary Do to a Credit or Fraud Model – and How Do You Defend?

Three attack families:

1. **Evasion** – craft an input that crosses the decision boundary while looking legitimate. Fraudsters test with small payments to probe the threshold.
2. **Poisoning** – inject malicious training data. A vendor supplying a "news sentiment" feed introduces biased tags.
3. **Model extraction / inversion** – query the model to recover its parameters or training data. An attacker with API access can reconstruct credit-score coefficients.

Finance-specific pressure points:

- APIs that return a credit score per query – extraction-friendly.
- Vendor-supplied features – poisoning-friendly.
- Voice-biometric KYC – evasion-friendly under synthetic audio.

Five defences:

- **Query budgeting** – rate-limit plus anomaly-score queries per requester.
- **Output coarsening** – return a bucket (A–E) instead of a raw score.
- **Adversarial training** – train on perturbed inputs so the boundary is robust.
- **Ensemble disagreement monitoring** – two independent models; divergence flagged for human review.
- **Input provenance tracking** – feature vectors stamped with source; poisoned vendor feeds can be traced and rolled back.

Regulatory angle. The EU AI Act's Article 15 requires "accuracy, robustness, and cybersecurity" for high-risk systems – effectively codifying defences 3 and 4.

Three attack families (evasion, poisoning, extraction) and five structural defences. The EU AI Act Article 15 codifies robustness and cybersecurity for high-risk systems; finance models are explicitly in scope.

(c) Joerg Osterrieder 2026

Three Historical Model Failures – and the Structural Lesson Each Left Behind

LTCM (1998).

- Convergence-trade models assumed normal returns with historical correlations.
- Russia default triggered a liquidity-driven correlation spike.
- Model did not misfire – it was solving the wrong problem (optimisation rather than survival).
- **Lesson:** stress-test correlation stability, not just level.

Copula CDO pricing (2007–2008).

- Gaussian copula with single correlation parameter priced senior tranches.
- In stress, tail correlations went to 1; "safe" tranches took 70 percent losses.
- The simple, tractable model dominated adoption precisely because of its tractability.
- **Lesson:** elegance and tractability can actively disqualify a model for tail-risk use.

Zillow iBuyer (2021).

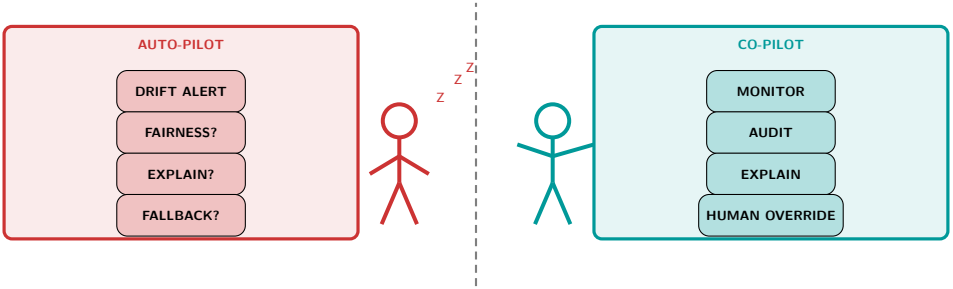
- Automated home-valuation model scaled buying to 7,000+ homes quarterly.
- Model calibrated on rising-market data; reversed in 2021.
- Zillow wrote down USD 881 m and exited iBuying entirely.
- **Lesson:** when the model controls the flow, small errors compound into structural losses.

The common thread.

1. All three models were mathematically correct given their assumptions.
2. All three assumptions failed because the world responded to the model's presence.
3. All three organisations lacked a *silent monitor* – a model or metric not used in decisions whose sole purpose is to flag regime change.

The take-away. A production model needs a watchdog that is not itself a production model.

LTCM, copula CDOs, Zillow iBuyer – three landmark failures, one structural gap. Each organisation lacked a silent monitor. A production model always needs a watchdog that is not itself a production model.



The interesting question is not whether to automate – it is where the human stays in the loop, and what evidence that presence leaves behind.