EWS Pipeline Documentation

Corporate Credit Early Warning System - Technical Workflow

Document Information
Project: Corporate Credit Early Warning System (12-month PD model)
Date: October 1, 2025
Purpose: Technical documentation of end-to-end modeling pipeline

---

Pipeline Overview

The EWS pipeline consists of **7 main stages** from raw data generation to production scoring:

```
1. Data Generation (Synthetic)
   ├─ gen_cohorts.py      → Monthly snapshots (backtest cohorts)
   ├─ gen_input.py        → Raw tables (financials, credit, cashflow, covenants)
   └─ gen_portfolio.py    → Scoring portfolio

2. Feature Engineering
   └─ feature_engineering.py → Financial ratios + behavioral features

3. Model Training
   └─ train_baseline.py   → LightGBM classifier + SHAP explainability

4. Calibration
   └─ calibrate.py        → Isotonic regression for probability calibration

5. Scoring
   └─ scoring.py          → Batch scoring with absolute thresholds

6. Validation & Testing
   ├─ backtest_monthly.py → Performance over 18 months
   ├─ calculate_psi.py    → Population Stability Index
   └─ plot_validation.py  → Validation dashboard

7. Monitoring & Stress Testing
   ├─ run_monitoring.py   → Production monitoring
   └─ stress_test.py      → Scenario stress testing
```

Stage 1: Data Generation

1.1 Generate Monthly Cohorts (`gen_cohorts.py`)

Purpose: Create monthly snapshots for backtesting (18 months: Jan 2024 → Jun 2025)

Inputs: None (synthetic generation)

Process:

1. Generate 10,000 customers per month
2. Assign sector (10 sectors: Manufacturing, Construction, Retail, etc.)
3. Assign credit grade (A–G) with realistic distribution:
   - Grade A: 82% (PD = 0.5%)
   - Grade B: 8% (PD = 1.0%)
   - Grade C–G: 10% (PD = 2–20%)
4. Apply sector multipliers (Construction × 1.3, Retail × 1.2, Tech × 0.8)
5. Add seasonal/temporal shocks (sine wave + linear trend)
6. Generate binary labels (default within 6M, 12M)

Outputs: - `data/processed/backtest_cohorts.parquet` (180,000 rows = 18 months × 10,000)

Key Parameters:

```
start = "2024-01-31"
end = "2025-06-30"
n_customers = 10,000
seed = 42
```

Sample Output:

customer_id	as_of_date	sector	grade	pd_12m	y_event_12m	ead	lgd
C00001	2024-01-31	MFG	A	0.0048	0	125,000	0.35
C00002	2024-01-31	CON	B	0.0132	1	580,000	0.42

1.2 Generate Raw Input Tables (`gen_input.py`)

Purpose: Generate 5 raw data tables simulating bank systems

Inputs: - Configuration: `Config` dataclass (customers, sectors, time windows) - As-of date: `2025-06-30` (snapshot date)

Process:

Step 1: Customer Master - 1,000 customers with sector, size bucket (SME 80%, Corp 20%) - Assign base financials (Revenue, EBITDA, Total Assets, Total Debt)

Step 2: Financial Statements (Quarterly) - 12 quarters of historical financials (Q3/2022 → Q2/2025) - Generate income statement & balance sheet items: - Revenue, COGS, Operating Expenses - EBITDA, Interest Expense, Tax - Total Assets, Total Debt, Shareholder Equity - Working Capital, Cash, Inventory - Apply industry-specific patterns

Step 3: Credit Behavior (Daily) - 180 days of credit line usage (lookback from as-of date) - Track daily: Limit, Outstanding, Utilization Rate, Days Past Due (DPD) - Simulate realistic payment patterns: - Good customers: DPD = 0–15 days - Risky customers: DPD spikes to 30–90+ days

Step 4: Cashflow Transactions (Daily) - 180 days of cash movements - Track: Total Inflow, Total Outflow, Net Cashflow - Detect negative days, volatility

Step 5: Covenant Monitoring - Track financial covenants (Debt/EBITDA ≤ 3.5x, ICR ≥ 1.5x) - Daily breach indicators (0/1)

Step 6: Labels - Binary outcome: `event_h12m` (default within 12 months after as-of date) - Label logic: - DPD ≥ 90 days for 30+ consecutive days → Default - Or: Utilization rate > 90% + covenant breach → Bump PD by 20–40%

Outputs (5 files): - `data/raw/fin_quarterly.parquet` (12K rows = 1K customers × 12 quarters) - `data/raw/credit_daily.parquet` (180K rows = 1K customers × 180 days) - `data/raw/cashflow_daily.parquet` (180K rows) - `data/raw/covenant.parquet` (180K rows) - `data/raw/labels.parquet` (1K rows = 1 label per customer)

Key Parameters:

```
n_customers = 1,000
n_quarters = 12 (financial history)
behavior_days = 180 (credit behavior window)
label_horizon_days = 365 (12 months forward)
asof_date = "2025-06-30"
```

1.3 Generate Scoring Portfolio (`gen_portfolio.py`)

Purpose: Create current portfolio for production scoring

Process: - Similar to `gen_input.py` but for production snapshot only - Generate customer master + financials + credit behavior at as-of date - No labels (production scoring doesn't have future outcomes yet)

Outputs: - `data/processed/portfolio_scored.csv` (production customer list)

Stage 2: Feature Engineering

Script: `feature_engineering.py`

Purpose: Transform raw tables into modeling-ready features

Inputs:

- 5 raw tables from Stage 1.2
- As-of date: `2025-06-30`
- Observation window: 180 days

Feature Categories (20+ features total):

A. Financial Ratios (from Quarterly Statements)

Leverage Ratios:

- `debt_to_ebitda` = Total Debt / EBITDA
- `debt_to_equity` = Total Debt / Shareholder Equity
- `debt_to_assets` = Total Debt / Total Assets

Coverage Ratios:

- `interest_coverage` = EBITDA / Interest Expense
- `ebitda_margin` = EBITDA / Revenue

Liquidity Ratios:

- `current_ratio` = Current Assets / Current Liabilities
- `cash_to_assets` = Cash / Total Assets

Operational Ratios:

- `asset_turnover` = Revenue / Total Assets
- `working_capital_to_revenue` = Working Capital / Revenue

Growth Indicators:

- `revenue_growth_qoq` = (Revenue_Q0 - Revenue_Q1) / Revenue_Q1
- `ebitda_growth_qoq` = (EBITDA_Q0 - EBITDA_Q1) / EBITDA_Q1

B. Credit Behavior (from Daily Credit Data)

Utilization Metrics (180-day window):

- `util_mean` = Average utilization rate
- `util_max` = Peak utilization
- `util_p95` = 95th percentile utilization
- `util_days_above_80` = # days with utilization > 80%

Delinquency Metrics:

- `dpd_mean` = Average days past due
- `dpd_max` = Maximum DPD in window
- `dpd_days_above_30` = # days with DPD > 30

- `dpd_max_streak` = Longest consecutive days with DPD ≥ 30

C. Cashflow Indicators (from Daily Cashflow)

- `cf_net_mean` = Average daily net cashflow
- `cf_negative_days` = # days with negative cashflow
- `cf_volatility` = Std dev of daily net cashflow

D. Covenant Breaches

- `cov_breach_days` = # days with covenant violation in 180-day window

E. Sector & Size Normalization

Z-score by (Sector, Size):

- For each numeric ratio, compute robust z-score:
  - `feature__zs_sector_size` = (value - median) / IQR
  - Grouped by (sector_code, size_bucket)
  - Uses median & IQR instead of mean/std for outlier robustness

Process Flow:

```
1. Load 5 raw tables
2. Filter data to observation window (as_of_date - 180 days → as_of_date)
3. Compute financial ratios (TTM = trailing 12 months)
4. Aggregate credit behavior (mean, max, percentiles over 180 days)
5. Aggregate cashflow metrics
6. Count covenant breaches
7. Normalize by sector & size (z-scores)
8. Join all features with labels
9. Drop missing values
10. Save as modeling dataset
```

Outputs: - `data/processed/model_features.parquet` (1 row per customer, 20+ feature columns)

Sample Row:

customer_id	debt_to_ebitda	interest_coverage	util_mean	dpd_max	event_h12m
C00001	2.3	4.5	0.45	0	0
C00002	5.8	1.2	0.87	45	1

Stage 3: Model Training

Script: `train_baseline.py`

Purpose: Train LightGBM classifier with calibration & explainability

Inputs: - `data/processed/model_features.parquet` - Target: `event_h12m` (binary: 0 = no default, 1 = default)

Process:

Step 1: Feature Selection

- Auto-select normalized features: `*__zs_sector_size` (prioritize z-scored features)
- If < 5 z-scored features available, use all numeric columns
- Drop: `customer_id`, `sector_code`, `size_bucket`, `event_h12m`

Step 2: Train-Test Split

- Split: 80% train, 20% test
- Stratified by target (maintain default rate balance)
- Random seed = 42 (reproducible)

Step 3: Train Base Model (LightGBM)

Hyperparameters:

```
objective = "binary"
metric = "auc"
num_leaves = 31
learning_rate = 0.05
n_estimators = 500
min_child_samples = 20
feature_fraction = 0.8
bagging_fraction = 0.8
bagging_freq = 5
early_stopping_rounds = 50
```

Training:

- Fit on train set with validation monitoring
- Early stopping if AUC doesn't improve for 50 rounds
- Save best iteration

Step 4: Probability Calibration

- Method: **Platt Scaling** (sklearn `CalibratedClassifierCV`)
- Calibrator: Sigmoid (logistic regression on LightGBM outputs)
- CV: 5-fold cross-validation on train set
- Purpose: Convert raw model scores → well-calibrated probabilities

Step 5: SHAP Explainability

- Compute **SHAP TreeExplainer** on test set (sample 100 customers)
- Generate global feature importance (mean |SHAP|)
- Save SHAP values for top 3 drivers per customer

Step 6: Evaluate Metrics (Test Set)

- **AUC-ROC**: Discrimination ability (target > 80%)
- **KS Statistic**: Max separation (TPR - FPR)
- **Brier Score**: Calibration error (target < 2%)
- **Precision-Recall AUC**: Performance on imbalanced data

Step 7: Define Thresholds

- **Red**: Top 5% highest risk (Red ≥ 95th percentile)
- **Amber**: Top 10% total (Amber ≥ 90th percentile, excluding Red)

Outputs:

- `artifacts/models/lgb_model.pkl` (trained LightGBM)
- `artifacts/models/calibrator.pkl` (Platt scaler)
- `artifacts/models/feature_names.json` (feature list)
- `artifacts/models/thresholds.json` (Red/Amber cutoffs)
- `artifacts/models/baseline_metrics.json` (AUC, KS, Brier)
- `artifacts/shap/shap_summary.csv` (feature importance)

Sample Metrics:

```json
{
  "AUC": 0.925,
  "KS": 0.783,
  "Brier": 0.0196,
  "PR_AUC": 0.161,
  "test_default_rate": 0.0137
}
```

Stage 4: Calibration

Script: `calibrate.py`

Purpose: Apply isotonic regression for better probability calibration

Inputs:

- `data/processed/scores_raw.csv` (raw model scores from Stage 3)
- Target: `event_h12m`

Process:

Isotonic Regression

- Fit: `IsotonicRegression(out_of_bounds="clip")`
- Input: Raw model score (0–1)
- Output: Calibrated probability (0–1)
- Constraint: Monotonic (higher score → higher probability)

Threshold Mapping

Two strategies available:

Strategy 1: Percentile-Based (Default from train_baseline.py)

- **Method**: Top 5% highest risk = Red, Top 10% total = Amber
- **Red**: PD ≥ 16.07% (95th percentile cutoff)
- **Amber**: PD ≥ 3.24% (90th percentile cutoff)
- **Green**: PD < 3.24%
- **Advantage**: Consistent alert volumes regardless of portfolio risk level
- **Use when**: Alert capacity is fixed (e.g., 500 analysts can handle 5% of portfolio)

Strategy 2: Absolute PD (Optional from calibrate.py)

- **Method**: Fixed PD thresholds
- **Red**: PD ≥ 20.0% (absolute cutoff)
- **Amber**: PD ≥ 5.0%
- **Green**: PD < 5.0%
- **Advantage**: Aligned with risk appetite, consistent with regulatory PD definitions
- **Use when**: Business has specific risk tolerance (e.g., "PD > 20% = unacceptable")

Current deployment: Uses **Strategy 1 (Percentile)** from `train_baseline.py`

EWS Score (0–100)

- Linear scaling: `score = prob_calibrated × 100`
- **Percentile mode**: Red 16–100, Amber 3–16, Green 0–3
- **Absolute mode**: Red 20–100, Amber 5–20, Green 0–5

Outputs:

- `artifacts/calibration/calibrator_isotonic.pkl` (isotonic model)
- `artifacts/calibration/thresholds.json` (absolute thresholds)
- `artifacts/calibration/mapping.csv` (raw score → calibrated PD)
- `data/processed/scores_calibrated.csv` (with tier assignments)

Sample Output:

customer_id	score_raw	prob_calibrated	score_ews	tier
C00001	0.0048	0.0052	5.2	Green
C00002	0.0654	0.0712	71.2	Red

Stage 5: Batch Scoring

Script: `scoring.py`

Purpose: Score production portfolio (customers without labels)

Inputs:

- `data/processed/model_features.parquet` (production features)
- `artifacts/models/lgb_model.pkl`
- `artifacts/models/calibrator.pkl`
- `artifacts/calibration/thresholds.json`

Process:

1. Load production customer features
2. Apply trained model → raw score
3. Apply calibrator → calibrated PD
4. Compute EWS score (0–100)
5. Assign tier (Red/Amber/Green) based on absolute thresholds
6. Map action recommendations:
   - **Green**: Routine monitoring, update financials on schedule
   - **Amber**: RM review ≤10 days, request management accounts, limit increases frozen
   - **Red**: Customer meeting ≤5 days, 13-week cashflow plan, watchlist, covenant tightening

Outputs:

- `artifacts/scoring/ews_scored_2025-06-30.csv`
  - Columns: `customer_id`, `prob_default_12m`, `score_ews`, `tier`, `action`
- `artifacts/scoring/thresholds_used.json` (for audit trail)

Sample Output:

customer_id	prob_default_12m	score_ews	tier	action
C00001	0.52%	5.2	Green	Theo dõi định kỳ; cập nhật BCTC đúng hạn.
C00125	3.8%	38.0	Amber	Soát xét RM ≤10 ngày; yêu cầu management accounts.

customer_id	prob_default_12m	score_ews	tier	action
C00847	12.5%	125	Red	Họp KH ≤5 ngày; lập kế hoạch dòng tiền 13 tuần.

Stage 6: Validation & Backtesting

6.1 Monthly Backtest (`backtest_monthly.py`)

Purpose: Test model performance over 18 months (Jan 2024 → Jun 2025)

Inputs:

- `data/processed/backtest_cohorts.parquet` (180K rows)
- Trained model + calibrator

Process:

1. For each month (18 iterations):
   - Filter cohort (10,000 customers)
   - Apply model → predict PD
   - Apply calibrator → calibrated PD
   - Assign tier (Red/Amber/Green)
   - Compare with actual labels (`y_event_12m`)
   - Compute metrics: AUC, KS, Brier, Precision, Recall
2. Aggregate monthly metrics

Outputs:

- `artifacts/backtest/monthly_metrics.csv` (18 rows)
  - Columns: `as_of_month`, `auc`, `ks`, `brier`, `precision`, `recall`, `amber_alert_rate`, `red_alert_rate`

Sample Output:

as_of_month	auc	ks	brier	precision	recall	amber_alert_rate	red_alert_rate
2024-01-31	0.834	0.601	0.0106	0.096	0.575	8.3%	4.2%
2024-02-29	0.819	0.582	0.0144	0.092	0.568	8.5%	4.3%

6.2 Population Stability Index (`calculate_psi.py`)

Purpose: Measure data drift across 18 months

Process:

1. Bucket scores into 10 deciles (baseline = first month)
2. For each subsequent month:
   - Compute distribution across same deciles
   - PSI = Σ (actual% - expected%) × ln(actual% / expected%)
3. PSI > 0.10 → Warning (data shifting)
4. PSI > 0.25 → Critical (recalibration needed)

Outputs:

- `artifacts/backtest/psi_monthly.csv` (17 rows, one per month vs. baseline)

Sample:

month	psi	status
2024-02-29	0.002	OK
2024-03-31	0.005	OK
2025-06-30	0.000	OK (synthetic → perfect stability)

6.3 Validation Dashboard (`plot_validation.py`)

Purpose: Generate visual validation report

Plots Generated (5 total):

1. **AUC & KS Trend Over Time** (`auc_ks_trend.png`)
   - Line charts: 18-month AUC & KS scores
   - Reference line: AUC = 75% (minimum threshold)
2. **Decile Calibration** (`decile_calibration.png`)
   - Bar chart: Predicted vs. Actual default rate by decile
   - Error bars: ±1 std dev
3. **Precision-Recall Curve** (`precision_recall_curve.png`)
   - Curve showing trade-off between precision & recall
   - Markers: Selected thresholds (Amber, Red)
4. **Alert Volume vs Threshold** (`alert_volume_vs_threshold.png`)
   - Sweep chart: Alert rate from 0.5% to 10% threshold
   - Dual axis: Precision & Recall
5. **Validation Dashboard** (`validation_dashboard.png`)
   - 2×2 layout combining: AUC/KS trend, Calibration, Threshold analysis, Summary metrics

Outputs:

- `artifacts/validation/plots/*.png` (5 files)
- `artifacts/validation/VALIDATION_REPORT_EN.md` (integrated report with plots)

Stage 7: Monitoring & Stress Testing

7.1 Production Monitoring (`run_monitoring.py`)

Purpose: Monthly tracking of production performance

Inputs:

- Current month's predictions + actuals (after 12 months maturity)
- Historical baseline metrics

Metrics Tracked:

1. **PSI** (data drift)
2. **AUC** (discrimination)
3. **Brier** (calibration)
4. **Alert Rate** (operational load)
5. **Actual Precision** (after labels available)

Alerts:

- PSI > 0.10 → Email warning
- AUC < 75% for 2 months → Recalibration trigger
- Alert rate > 15% → Threshold adjustment

Outputs:

- `artifacts/monitoring/monitoring_YYYYMMDD_HHMMSS.json` (timestamped)
- `artifacts/monitoring/monitoring_metrics.csv` (cumulative)

7.2 Stress Testing (`stress_test.py`)

Purpose: Test model under crisis scenarios

Scenarios (from `stress_scenarios.yaml`):

1. **Baseline**: Current conditions (no shock)
2. **Mild Recession**: Revenue -15%, Debt +10%
3. **Severe Recession**: Revenue -30%, Debt +25%, Liquidity -20%
4. **Sector Shock**: Construction sector EBITDA -40%
5. **Liquidity Crisis**: Utilization +30pp, Cashflow volatility ×3

Process:

1. Load base features
2. Apply scenario shocks to features
3. Re-score with model
4. Compare tier migrations (Green → Amber → Red)

Outputs:

- `artifacts/stress_testing/stress_results.csv`
    - Columns: `scenario`, `baseline_red%`, `stressed_red%`, `migration_rate`
- `artifacts/stress_testing/STRESS_TEST_NOTE.md` (report with waterfall charts)

Sample Result:

Scenario	Baseline Red%	Stressed Red%	Migration
Mild Recession	4.2%	7.8%	+85%
Severe Recession	4.2%	15.3%	+264%

Data Flow Diagram

```
┌──────────────────────────────────────────────────────────────────┐
│ Stage 1: Data Generation                                         │
│ ┌─────────────┐  ┌──────────────┐  ┌───────────────┐          │
│ │gen_cohorts  │  │ gen_input    │  │gen_portfolio  │          │
│ │(backtest)   │  │ (5 raw tables│  │(production)   │          │
│ └──────┬──────┘  └──────┬───────┘  └───────┬───────┘          │
└────────┼────────────────┼──────────────────┼─────────────────────┘
         │                │                  │
         v                v                  v
   backtest_cohorts   raw/*.parquet   portfolio_scored.csv
         │                │                  │
         └────────┬───────┴──────────────────┘
                  v
┌──────────────────────────────────────────────────────────────────┐
│ Stage 2: Feature Engineering                                     │
│ ┌──────────────────────────────────────────────────────┐      │
│ │ feature_engineering.py                               │      │
│ │ • Compute 20+ features (ratios, behavior, CF)        │      │
│ │ • Z-score normalization by sector/size             │      │
│ └─────────────────────────┬────────────────────────────┘      │
└───────────────────────────┼──────────────────────────────────────┘
                            v
                 model_features.parquet
                            │
         ┌──────────────────┼──────────────────┐
         v                  v                  v
    (train set)        (test set)      (production set)
         │                  │                  │
         v                  v                  │
┌──────────────────────────────────────────────┼───────────────────┐
│ Stage 3: Model Training                      │                   │
│ ┌──────────────────────────┐                 │                   │
```

```
│ │ train_baseline.py        │            │                       │
│ │ • LightGBM + Platt       │            │                       │
│ │ • SHAP explainability    │            │                       │
│ └───────┬──────────────────┘            │                       │
└─────────┼───────────────────────────────┼───────────────────────┘
          v                               │
    lgb_model.pkl                         │
    calibrator.pkl                        │
    thresholds.json                       │
          │                               │
          v                               │
┌─────────────────────────────────────────┼───────────────────────┐
│ Stage 4: Calibration                    │                       │
│ ┌──────────────────────────┐            │                       │
│ │ calibrate.py             │            │                       │
│ │ • Isotonic regression    │            │                       │
│ │ • Absolute thresholds    │            │                       │
│ └───────┬──────────────────┘            │                       │
└─────────┼───────────────────────────────┼───────────────────────┘
          v                               │
    calibrator_isotonic.pkl               │
    scores_calibrated.csv                 │
          │                               │
          └───────────────┬───────────────┘
                          v
┌─────────────────────────────────────────────────────────────────┐
│ Stage 5: Scoring                                                │
│ ┌───────────────────────────────────────────────────┐           │
│ │ scoring.py                                         │           │
│ │ • Batch prediction                                 │           │
│ │ • Tier assignment (Red/Amber/Green)                │           │
│ │ • Action recommendations                           │           │
│ └───────────────────────┬───────────────────────────┘           │
└─────────────────────────┼───────────────────────────────────────┘
                          v
                 ews_scored_YYYY-MM-DD.csv
                          │
         ┌────────────────┴────────────────┐
         v                                 v
┌──────────────────────────┐   ┌─────────────────────────────────┐
│ Stage 6: Validation       │   │ Stage 7: Monitoring/Stress       │
│ • backtest_monthly.py     │   │ • run_monitoring.py              │
│ • calculate_psi.py        │   │ • stress_test.py                 │
│ • plot_validation.py      │   │                                  │
│                           │   │                                  │
│ Outputs:                  │   │ Outputs:                         │
│ • monthly_metrics.csv     │   │ • monitoring_*.json              │
│ • psi_monthly.csv         │   │ • stress_results.csv             │
│ • validation_*.png        │   │ • STRESS_TEST_NOTE.md            │
```

```
│ • VALIDATION_REPORT.md │   │                          │
└────────────────────────┘   └──────────────────────────┘
```

Key Files & Artifacts

Input Data

```
data/
├── raw/                           # Stage 1.2 outputs
│   ├── fin_quarterly.parquet      # 12K rows (1K × 12Q)
│   ├── credit_daily.parquet       # 180K rows (1K × 180d)
│   ├── cashflow_daily.parquet     # 180K rows
│   ├── covenant.parquet           # 180K rows
│   └── labels.parquet             # 1K rows
├── processed/
│   ├── backtest_cohorts.parquet   # 180K rows (backtest)
│   ├── model_features.parquet     # 1K rows (20+ features)
│   ├── scores_raw.csv             # Raw model outputs
│   ├── scores_calibrated.csv      # Calibrated PD + tiers
│   └── portfolio_scored.csv       # Production portfolio
```

Model Artifacts

```
artifacts/
├── models/
│   ├── lgb_model.pkl              # Trained LightGBM (500 trees)
│   ├── calibrator.pkl             # Platt scaler
│   ├── feature_names.json         # 20 feature list
│   ├── thresholds.json            # Red/Amber cutoffs
│   └── baseline_metrics.json      # AUC, KS, Brier
├── calibration/
│   ├── calibrator_isotonic.pkl    # Isotonic regression
│   ├── thresholds.json            # Absolute PD thresholds
│   └── mapping.csv                # Score → PD mapping
├── shap/
│   ├── summary.json               # Global feature importance
│   ├── feature_importance.csv     # SHAP rankings
│   └── top_drivers_per_customer.csv  # Top 3 drivers per alert
├── backtest/
│   ├── monthly_metrics.csv        # 18-month performance
│   ├── psi_monthly.csv            # Stability tracking
│   ├── threshold_sweep.csv        # Precision/Recall curves
│   └── BACKTEST_REPORT.html       # Quarto report
├── validation/
│   ├── plots/
│   │   ├── auc_ks_trend.png
│   │   ├── decile_calibration.png
│   │   ├── precision_recall_curve.png
```

```
│   │   ├── alert_volume_vs_threshold.png
│   │   └── validation_dashboard.png
│   └── VALIDATION_REPORT_EN.md       # English validation report
├── scoring/
│   ├── ews_scored_2025-06-30.csv     # Scored customers
│   └── thresholds_used.json          # Audit trail
├── monitoring/
│   ├── monitoring_YYYYMMDD.json      # Monthly snapshots
│   └── monitoring_metrics.csv        # Cumulative tracking
└── stress_testing/
    ├── stress_results.csv            # Scenario results
    ├── stress_scenarios.yaml         # Scenario definitions
    └── STRESS_TEST_NOTE.md           # Stress test report
```

Execution Guide

Full Pipeline (from scratch)

```
# 1. Generate data
python src/gen_data/gen_cohorts.py --start 2024-01-31 --end 2025-06-30 --n 10000
python src/gen_data/gen_input.py --output-dir data/raw --n 1000
python src/gen_data/gen_portfolio.py --output data/processed/portfolio_scored.csv

# 2. Feature engineering
python src/modeling/feature_engineering.py \
    --raw-dir data/raw \
    --asof-date 2025-06-30 \
    --output data/processed/model_features.parquet

# 3. Train model
python src/modeling/train_baseline.py \
    --input data/processed/model_features.parquet \
    --target event_h12m \
    --outdir artifacts/models

# 4. Calibration (Optional - only if switching to absolute thresholds)
# Main pipeline uses percentile thresholds from train_baseline.py
# Run this step ONLY if business requires absolute PD cutoffs

# First, extract raw scores from trained model:
python src/make_scores_raw.py \
    --features data/processed/feature_ews.parquet \
    --model artifacts/models/model_lgbm.pkl \
    --y-col event_h12m \
    --out data/processed/scores_raw.csv

# Then apply isotonic calibration with absolute thresholds:
python src/calibrate.py \
```

```
    --input data/processed/scores_raw.csv \
    --y-col event_h12m \
    --score-col score_raw \
    --red-thr 0.20 \
    --amber-thr 0.05 \
    --outdir artifacts/calibration

# Note: By default, train_baseline.py already creates calibrated scores
# with percentile-based thresholds (Red=top 5%, Amber=top 10%)

# 5. Batch scoring
python src/scoring.py \
    --features data/processed/portfolio_scored.csv \
    --model artifacts/models/lgb_model.pkl \
    --thresholds artifacts/calibration/thresholds.json \
    --asof 2025-06-30 \
    --outdir artifacts/scoring

# 6. Validation
python src/backtest/backtest_monthly.py
python src/backtest/calculate_psi.py
python src/plot_validation.py all

# 7. Monitoring
python src/run_monitoring.py --asof 2025-06-30
python src/stress_test.py --scenarios artifacts/stress_testing/stress_scenarios.yaml
```

Quick Validation Dashboard Only

```
python src/plot_validation.py dashboard
```

Performance Benchmarks

Model Metrics (Test Set)

- **AUC**: 92.5% (target > 80%) ✓✓ (excellent)
- **KS**: 78.3% (excellent separation)
- **Brier Score**: 1.96% (target < 2%) ✓
- **PR-AUC**: 16.1% (reasonable for 1.37% default rate)

Operational Metrics (Backtest Average)

- **Amber Alert Rate**: 8.3% (830 customers/month)
- **Amber Precision**: 9.6% (1 in 10 alerts is real default)
- **Amber Recall**: 57.5% (catches 57.5% of defaults)
- **Red Alert Rate**: 4.2% (420 customers/month)

- **False Positive Rate**: 90.4% (9 out of 10 Amber alerts are false)
- **Missed Defaults**: 42.5% (not flagged by system)
- **Workload**: ~15 FTE needed (5 for Amber, 10 for Red)

Stability (18 months backtest)

- **PSI**: 0.00 (synthetic data, perfect stability)
- **AUC Range**: 79.2% – 85.9% (±3.5% variation)
- **AUC Mean (backtest)**: 82.3% (slightly lower than test set due to time decay)
- **No degradation** observed in test period

Key Assumptions & Limitations

Data Quality

1. **Synthetic Data**: Test data is artificially generated
   - No real-world noise (missing data, reporting delays, data errors)
   - PSI = 0 is unrealistic (real production will have drift)
   - **Mitigation**: Run 6-month pilot with real data
2. **Label Quality**: Binary default definition (DPD ≥ 90 for 30 days)
   - May not capture all credit deterioration signals
   - No restructuring/forbearance cases simulated

Model Scope

1. **No Segmentation**: Single model for all sectors/sizes
   - May underperform in specific industries (Construction, Retail)
   - **Recommendation**: Build sector-specific models if performance gaps found
2. **12-Month Horizon Only**: Fixed 12-month PD
   - No 6-month or 24-month variants
   - May miss near-term acute risks or long-term structural issues
3. **Feature Coverage**: 20 features (financial + behavioral)
   - No macroeconomic variables (GDP, interest rates, FX)
   - No qualitative factors (management quality, industry trends)
   - No external data (credit bureau, industry benchmarks)

Operational Constraints

1. **High False Positive Rate**: 90% of Amber alerts are false
   - Risk of alert fatigue
   - **Mitigation**: Clear workflow, Red priority, periodic threshold review
2. **Missed Defaults**: 42.5% not caught by Amber/Red
   - System is supplementary, not standalone
   - **Mitigation**: Combine with quarterly credit reviews

Appendix: Technical Details

LightGBM Hyperparameters

```
params = {
    'objective': 'binary',
    'metric': 'auc',
    'num_leaves': 31,
    'learning_rate': 0.05,
    'n_estimators': 500,
    'min_child_samples': 20,
    'feature_fraction': 0.8,
    'bagging_fraction': 0.8,
    'bagging_freq': 5,
    'verbosity': -1,
    'early_stopping_rounds': 50
}
```

Feature Importance (Top 10)

Rank	Feature	SHAP Value	Description
1	`covenant_breach_cnt_180d`	0.710	Covenant violation count (180 days)
2	`delta_dso_qoq`	0.586	Days Sales Outstanding QoQ change
3	`dpo`	0.530	Days Payable Outstanding
4	`icr_ttm`	0.504	Interest Coverage Ratio (TTM)
5	`%util_p95_60d`	0.503	95th percentile utilization (60 days)
6	`debt_to_ebitda`	0.471	Leverage ratio (Debt/EBITDA)
7	`dso`	0.359	Days Sales Outstanding
8	`dpd_trend_180d`	0.326	DPD trend direction (180 days)
9	`dpd_max_180d`	0.320	Maximum days past due (180 days)
10	`doh`	0.279	Days on Hand (inventory)

Top 3 features account for **40.9%** of model explanatory power (total SHAP = 1.826).

Note: All features are z-score normalized by sector & size (`__zs_sector_size` suffix).