Reliability of Explainable AI (XAI)

October 19, 2025

1 Abstract

Problem Statement: Quantifying the *reliability* of feature-attribution Explainable AI (XAI) for **software defect prediction**—i.e., whether explanations truly reflect how the trained model behaves.

Proposed Methodology: We introduce a general evaluation framework that compares explainer rankings to a *grounded behavioral signal* derived from the model itself. Concretely, we (i) estimate *true loss sensitivity* via small ε-perturbations and finite differences on binary cross-entropy, (ii) assess **Gradient–Loss Reliability (GLR)** using Spearman rank correlation between SHAP importances and normalized sensitivity, (iii) validate *action-consistency* with **ε-Hit@k**, and (iv) quantify probability honesty with **Expected Calibration Error (ECE)**. A composite **ReliabilityScore** summarizes these facets.

Application: We demonstrate the framework on the **NASA PROMISE CM1** dataset (module-level defect prediction) using a **Random Forest** classifier with **SHAP (TreeExplainer)**. The pipeline reports discrimination (AUC, F1, Precision, Recall, Brier), GLR, ε-Hit@k, ECE, and the composite ReliabilityScore, alongside visualization (top mean$|\phi|$ features, GLR histograms, calibration curves).

Note: Software defect prediction aims to identify defect-prone modules from static code metrics (size, complexity, Halstead measures). Reliable explanations help practitioners prioritize inspections and reason about model decisions in safety- and cost-sensitive engineering workflows.

2 Introduction

Problem. Machine Learning (ML) is widely used for **software defect prediction** from static code metrics. However, many high-performing models (e.g., Random Forests, Neural Networks) behave as *black boxes*: they output probabilities without revealing *why.* In software engineering workflows—where inspection budgets, safety, and maintenance decisions depend on trust—engineers need transparent, dependable reasoning behind predictions.

Role of XAI. Explainable AI (XAI) provides per-feature attributions that indicate which metrics (e.g., LOC, cyclomatic complexity, Halstead measures) drive a module's defect risk. In this work we use **SHAP (TreeExplainer)** to obtain local and global importance over code metrics.

Issue with Current Explanations. Explanations can be *unstable* across folds, data slices, or minor distribution shifts, and they may not faithfully reflect the model's *true behavior* near a prediction. Without a standard reliability check, feature attributions risk being misleading or overconfident.

Contribution. We propose a **structured reliability evaluation** for feature-attribution XAI on defect prediction:

- A *behavior-grounded signal* via small ε-perturbations and binary cross-entropy to estimate **true loss sensitivity**.

- **Gradient–Loss Reliability (GLR)**: Spearman rank alignment between SHAP importances and normalized true-loss sensitivities.
- **ε-Hit@k**: an action-consistency test that checks whether SHAP's top-k features also cause the largest loss increases under tiny nudges.
- **Calibration (ECE)**: a probability honesty check complementary to discrimination metrics.
- A composite **ReliabilityScore** aggregating GLR, ε-Hit@k, and $1-$ECE for a single, interpretable summary.

We demonstrate this framework on the **NASA PROMISE CM1** dataset with a **Random Forest** model and SHAP explanations, reporting discrimination (AUC, F1, Precision, Recall, Brier), explanation reliability (GLR, ε-Hit@k), and calibration (ECE), alongside diagnostic visualizations.

3 Literature Survey

- **Emphasis gap.** Most prior work prioritizes *accuracy* or raw *interpretability*, while *reliability* is rarely stress-tested. Existing reliability implementations are largely *domain-specific*, often framed as a single generalized k-fold SHAP/LIME pipeline.
- **Under-specified dimensions.** Traditional approaches under-specify core explainability dimensions: explanations *vary with train/test splits* (hurting generalization), can *disagree with model attributions* (low concordance), and may *lack stability.*

Title of the Paper	Published In	Significance	Limitation
LCNN: Lightweight CNN for Software Defect Feature Identification Using Explainable AI	IEEE Access 2024	Used SHAP and LIME to interpret CNN-based predictions, highlighting reproducibility of explanations in deep models.	No systematic reliability assessment (e.g., stability, concordance); limited to visualization of explanations.
A Methodology for Reliability Analysis of Explainable AI: Application to Endocrinology Diseases	IEEE Access 2024	Introduced a generalized framework with *Generalizability*, *Concordance*, and *Stability* as reliability metrics for SHAP/LIME.	Domain-specific evaluation (healthcare); ignores local sensitivity / perturbation-based reliability.
Understanding Software Defect Prediction Through eXplainable Neural Additive Models (XNAMs)	IEEE Access 2025	Designed inherently interpretable models, reducing reliance on post-hoc explanations for consistency.	Focused on interpretability, not reliability; no cross-fold stability or perturbation analysis.

Table 1: Snapshot of recent work: strengths and reliability gaps.

Across these studies, there is still no universal standard for evaluating *explanation reliability*; cross-fold stability, concordance with model behavior, and perturbation-grounded checks remain under-explored and inconsistently reported.

Current Scenario

4 Methodology

We propose to combine XAI with k-fold cross-validation and evaluate using three metrics: **Generalizability**, **Concordance**, and **Stability**. These are aggregated into a **Global Reliability Score**.

4.1 Step 1: XAI with k-Fold Cross Validation

1. Split dataset into k folds.
2. For each fold:
   - Train the model on $k-1$ folds.
   - Test on the remaining fold.
   - Generate feature importance via SHAP or LIME.
3. Collect k sets of feature importances across folds.

This captures the variability of feature importance across different splits.

4.2 Step 2: Define Key Metrics

Generalizability. Do train vs test explanations rank features similarly (per fold)?

Let $\mathbf{r}^{(j)}_{\text{train}}$ and $\mathbf{r}^{(j)}_{\text{test}}$ be the rank vectors (of mean |SHAP| importances) for fold j over the same n features. Compute Spearman's ρ and rescale to $[0, 1]$:

$$\rho^{(j)}_{\text{gen}} = \text{Spearman}\Big(\mathbf{r}^{(j)}_{\text{train}}, \mathbf{r}^{(j)}_{\text{test}}\Big), \qquad G = \frac{1}{k}\sum_{j=1}^{k} \tilde{\rho}^{(j)}_{\text{gen}}, \quad \tilde{\rho} = \frac{\rho+1}{2} \in [0,1].$$

Concordance. Does the XAI importance agree with the model's internal (Gini/MDI) importance?

Let $\mathbf{r}^{(j)}_{\text{SHAP}}$ and $\mathbf{r}^{(j)}_{\text{Gini}}$ be rank vectors for fold j. Use **Spearman** (not Pearson), rescaled to $[0, 1]$:

$$\rho^{(j)}_{\text{con}} = \text{Spearman}\Big(\mathbf{r}^{(j)}_{\text{SHAP}}, \mathbf{r}^{(j)}_{\text{Gini}}\Big), \qquad C = \frac{1}{k}\sum_{j=1}^{k} \tilde{\rho}^{(j)}_{\text{con}}, \quad \tilde{\rho} = \frac{\rho+1}{2}.$$

Stability. Are explanations consistent across folds?

Let $\{\mathbf{s}^{(j)}\}_{j=1}^{k}$ be the per-fold importance vectors (e.g., mean |SHAP| on *train*). Compute mean pairwise **Spearman** across all fold pairs, rescaled to $[0, 1]$:

$$S = \frac{2}{k(k-1)} \sum_{1\le p<q\le k} \tilde{\rho}\Big(\text{rank}\big(\mathbf{s}^{(p)}\big), \text{rank}\big(\mathbf{s}^{(q)}\big)\Big), \quad \tilde{\rho} = \frac{\rho+1}{2}.$$

4.3 Step 3: Global Reliability Score

We combine the three metrics via their **arithmetic mean**:

$$\text{Reliability} = R = \frac{G+C+S}{3},$$

with $R \in [0, 1]$; larger values indicate more reliable explanations.

5 Implementation

5.1 Generalizability

Generalizability measures how well the feature importance derived from the training data generalizes to unseen test data. It is quantified using the **Spearman rank correlation coefficient** between the ranks of feature importances obtained from the training and test sets in each fold of cross-validation.

Formula. Let $R^{(j)}_{train}$ and $R^{(j)}_{test}$ be the rank vectors of feature importances (e.g., mean absolute SHAP values) for fold j. The Spearman correlation coefficient ρ for fold j is given by:

$$\rho^{(j)} = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\,(n^2-1)}$$

where

- n is the number of features,
- $d_i = \text{rank}_{train}(f_i) - \text{rank}_{test}(f_i)$ is the difference between the training and testing ranks of feature f_i.

The overall generalizability score across k folds is the average:

$$G = \frac{1}{k}\sum_{j=1}^{k} \rho^{(j)}$$

Example. Consider $n = 4$ features with the following ranks:

Feature	Train Rank	Test Rank	d_i	d_i^2
f_1	1	1	0	0
f_2	2	3	-1	1
f_3	3	2	1	1
f_4	4	4	0	0

Here, $\sum d_i^2 = 2$. Substituting in the formula:

$$\rho = 1 - \frac{6 \cdot 2}{4 \cdot (4^2 - 1)} = 1 - \frac{12}{60} = 0.8$$

Thus, the generalizability for this fold is 0.8, indicating a high consistency between train and test feature rankings.

5.2 Concordance

Concordance measures the agreement between feature importance obtained from post-hoc explainability methods and those derived directly from the learning algorithm. In our case, we compare **SHAP-based feature importances** with the **Gini importance** (Mean Decrease in Impurity, MDI) computed from tree-based models such as Random Forest. The agreement is quantified using the **Spearman rank correlation coefficient**.

Formula. Let $R_{\text{SHAP}}^{(j)}$ and $R_{\text{Gini}}^{(j)}$ be the rank vectors of feature importances obtained by SHAP and Gini index, respectively, for fold j. The Concordance for fold j is:

$$C^{(j)} = \rho\left(R_{\text{SHAP}}^{(j)}, R_{\text{Gini}}^{(j)}\right)$$

where ρ is the Spearman correlation coefficient defined as:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\,(n^2-1)}$$

with

- n being the total number of features,
- $d_i = \text{rank}_{\text{SHAP}}(f_i) - \text{rank}_{\text{Gini}}(f_i)$ being the difference in rank of feature f_i under SHAP and Gini importance.

The overall concordance score across k folds is then:

$$C = \frac{1}{k}\sum_{j=1}^{k} C^{(j)}$$

Example. Suppose we have four features with the following importance ranks:

Feature	SHAP Rank	Gini Rank	d_i	d_i^2
f_1	1	2	-1	1
f_2	2	1	1	1
f_3	3	3	0	0
f_4	4	4	0	0

Here, $\sum d_i^2 = 2$ and $n = 4$. Substituting:

$$\rho = 1 - \frac{6 \cdot 2}{4 \cdot (4^2 - 1)} = 1 - \frac{12}{60} = 0.8$$

Thus, the concordance for this fold is 0.8, indicating strong alignment between SHAP and Gini-based explanations.

5.3 Stability

Stability quantifies how consistent the *explanations* (feature-importance vectors) are across different resamples/folds of the data. Given one importance vector per fold (e.g., mean absolute SHAP values on the *train* split of each fold), we compute the **Spearman rank correlation** for every pair of folds and average these correlations.

Setup. Let F be the number of folds and let $\mathbf{s}^{(p)} \in \mathbb{R}^n$ denote the vector of feature importances for fold p over n common features. Let $\mathbf{r}^{(p)} = \text{rank}(\mathbf{s}^{(p)})$ be the rank vector ($1 =$ most important). Then the Spearman correlation between folds p and q is

$$\rho\left(\mathbf{r}^{(p)}, \mathbf{r}^{(q)}\right) = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\,(n^2-1)}, \quad d_i = r_i^{(p)} - r_i^{(q)}.$$

Definition. The Stability score is the mean pairwise Spearman correlation:

$$S \;=\; \frac{2}{F(F-1)} \sum_{1 \le p < q \le F} \rho\left(\mathbf{r}^{(p)}, \mathbf{r}^{(q)}\right).$$

Example. Assume $F = 3$ folds and $n = 4$ features. The Spearman correlations between fold pairs are $\rho_{12} = 0.86$, $\rho_{13} = 0.78$, $\rho_{23} = 0.80$. Then

$$S \;=\; \frac{2}{3 \cdot 2}\,(0.86 + 0.78 + 0.80) = \frac{1}{3} \cdot 2.44 \approx 0.813.$$

5.4 Reliability

Reliability provides an overall measure of trustworthiness by aggregating the three criteria: Generalizability (G), Concordance (C), and Stability (S). Since all three values lie in the range $[0, 1]$, we define the reliability score as their arithmetic mean:

$$R = \frac{G + C + S}{3}$$

where

- G is the generalizability score,
- C is the concordance score,
- S is the stability score.

Example. Suppose we obtained:

$$G = 0.75, \quad C = 0.80, \quad S = 0.60$$

Then:

$$R = \frac{0.75 + 0.80 + 0.60}{3} = \frac{2.15}{3} \approx 0.717$$

Thus, the overall reliability of the explanations is 0.717, indicating moderately strong reliability.

6 Results

Table 2: Reliability metrics (scaled to $[0, 1]$).

Metric	**Value**
Generalizability (Spearman train vs. test)	0.9866
Concordance (SHAP vs. MDI, Spearman)	0.9154
Stability (Pairwise Spearman across folds)	0.7788
Reliability Index $\left(\frac{G+C+S}{3}\right)$	0.9468

Insights.

- **Generalizability (0.9866):** Train–test explanation ranks are almost identical, indicating the explanation pattern holds strongly on unseen data.
- **Concordance (0.9154):** High agreement between SHAP and model-internal Gini (MDI) importances; post-hoc XAI is consistent with the model's own attributions.
- **Stability (0.7788):** Explanation vectors are fairly consistent across folds (mean pairwise Spearman), with some expected variability from resampling.
- **Reliability Index (0.9468):** Averaging G, C, and S yields a strong overall reliability—explanations are robust, model-aligned, and reasonably stable.

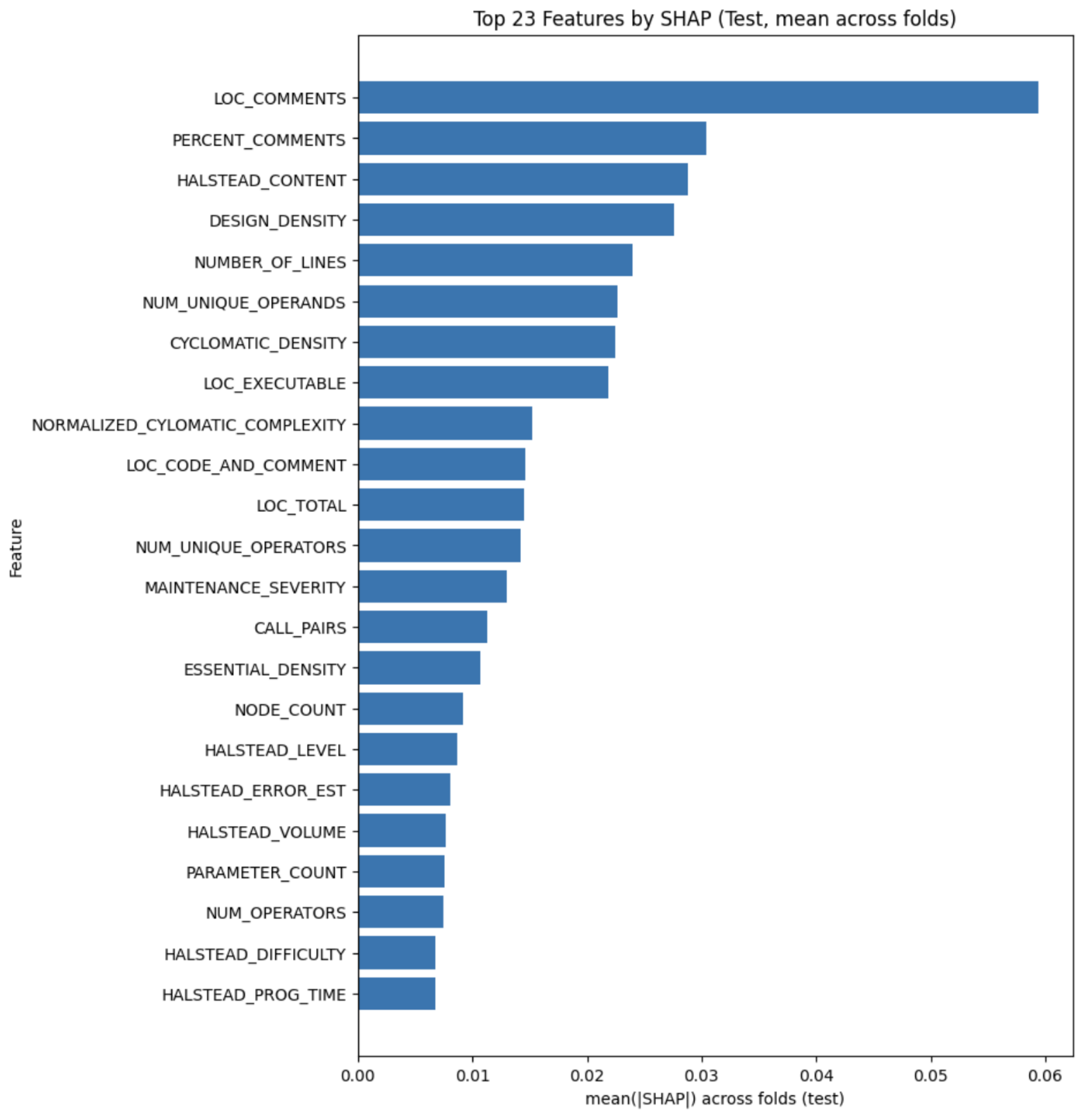

Figure 1: Visualization of feature importance scores using SHAP (SHapley Additive exPlanations) for the software defect dataset. Each bar corresponds to the average absolute SHAP value of a feature, representing its global contribution to the model's predictive performance. Features with longer bars have a greater influence on the model's decision-making process. This visualization highlights which software metrics (e.g., lines of code, number of developers, modification entropy) play the most critical role in predicting defect proneness.

Explanation. The SHAP method is grounded in cooperative game theory, where each feature is considered a "player" contributing to the final prediction. The SHAP value for a feature quantifies its marginal contribution averaged across all possible feature coalitions. By plotting the mean absolute SHAP values, we obtain a ranking of features by importance. This allows us to interpret the model globally, identifying the most influential factors driving defect prediction. For instance, in many defect datasets, features like *code churn*, *developer experience*, and *complexity metrics* emerge as dominant contributors, aligning with empirical software engineering studies.

Proposed Method

7 Framework

The proposed framework evaluates the **reliability of XAI** for a **software defect prediction** model trained with a Random Forest, using SHAP explanations and three reliability checks: *GLR (rank alignment)*, *ε-Hit@k* (perturbation validity), and *ECE* (probability calibration).

7.1 Dataset

- Software defect dataset with binary target `Defective` (e.g., `CM1.csv`).

7.2 Data Preprocessing (as in code)

(a) Load CSV; split into features X and label $y =$ `Defective`.

(b) **Drop constant columns**: remove any feature with only one unique value.

(c) **Missing values**: if present, apply forward fill then back fill.

(d) *Note:* No scaling/standardization is applied in the code.

7.3 Model Training & Evaluation

(a) **Cross-validation**: StratifiedKFold with *n_splits* folds (shuffle, fixed seed).

(b) **Class imbalance (train only)**: SMOTE with adaptive k-neighbors:

$$k = \min(5,\ \max(1,\ \#\text{minority} - 1)).$$

(c) **Classifier**: RandomForestClassifier

$\texttt{n\_estimators} = 600, \quad \texttt{min\_samples\_leaf} = 2, \quad \texttt{class\_weight} = \text{“balanced\_subsample”}, \quad \texttt{random\_sta}$

(d) **Per-fold metrics on test**: AUC, F1, Precision, Recall, Brier score.

(e) Aggregate means across folds: $\overline{\text{AUC}}, \overline{\text{F1}}, \overline{\text{Precision}}, \overline{\text{Recall}}, \overline{\text{Brier}}$.

7.4 Explainable AI (XAI) with SHAP (as in code)

- Use **TreeExplainer** with a background sample of $\min(256,\ |X_{\text{train}}|)$, `feature_perturbation=intervent` `model_output=probability`.
- Compute per-sample SHAP values on test; use absolute values $|\phi|$.
- Compute global importance as mean of $|\phi|$ across test samples.

7.5 True Loss Sensitivity via Finite Differences (as in code)

(a) Binary cross-entropy on probability p:

$$\ell(p, y) = -\Big(y \log p + (1 - y) \log(1 - p)\Big).$$

(b) For each test sample $x \in \mathbb{R}^d$ and each feature j, approximate

$$g_j \approx \left|\frac{\partial \ell}{\partial x_j}\right| \approx \frac{\left|\ell(p(x+\varepsilon e_j), y) - \ell(p(x-\varepsilon e_j), y)\right|}{2\varepsilon},$$

where $p(\cdot)$ is `predict_proba` class-1 output of the RF and e_j is the unit vector.

(c) Normalize per sample:

$$\tilde{g}_j = \frac{g_j}{\sum_{k=1}^{d} g_k + \epsilon}.$$

7.6 Gradient–Loss Reliability (GLR)

(A) For each test sample, take vectors S (absolute SHAP) and G (normalized sensitivities $\tilde{g}$).

(B) Convert to ranks and compute **Spearman** correlation ρ between S and G:

$$\rho = 1 - \frac{6\sum_{j=1}^{d}(r_j^S - r_j^G)^2}{d(d^2-1)}.$$

(C) **GLR score** is the mean of ρ over all test samples (ignoring NaN).

7.7 ε-Hit@k (Perturbation Validity Check, as in code)

(A) For each selected test sample (random $\sim 15\%$ per fold until a global budget `max_eps_samples`):

- (a) Let Top-M_{SHAP} be indices of the $M = \min(\texttt{m\_limit}, d)$ largest $|\phi|$.
- (b) Let T_k be SHAP top-k within those M.
- (c) For each $j \in$ Top-M_{SHAP}, perturb $x_j \leftarrow x_j + \varepsilon$; compute

$$\Delta\ell_j = \ell\big(p(x+\varepsilon e_j), y\big) - \ell\big(p(x), y\big).$$

- (d) Let D_k be indices of the top-k features by $\Delta\ell_j$.

(B) Define per-sample hit:

$$\text{Hit@k(sample)} = \mathbf{1}\{\, T_k \cap D_k \neq \emptyset \,\}.$$

(C) **ε-Hit@k** is the mean of these hits over all sampled test rows.

7.8 Calibration (ECE, as in code)

(a) Expected Calibration Error with B bins:

$$\text{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\, \text{acc}(b) - \text{conf}(b) \,\right|,$$

where conf(b) is the mean predicted probability and acc(b) is the empirical positive rate in bin b.

7.9 Composite Reliability Score (as in code)

- Rescale GLR to $[0, 1]$: $\rho' = \frac{\rho+1}{2}$.
- Combine:

$$\text{ReliabilityScore} = \text{mean}\Big(\rho',\ \text{Hit@k},\ \max\{0,\, 1-\text{ECE}\}\Big).$$

7.10 Visualization (as produced by code)

- **Top features by mean(—SHAP—)** across folds (horizontal bar plot).
- **GLR histogram** showing distribution of per-sample Spearman ρ.
- **Calibration curve** (predicted probability vs. fraction of positives) with ECE in title.
- (Optional) **SHAP Beeswarm snapshot** for the last fold.

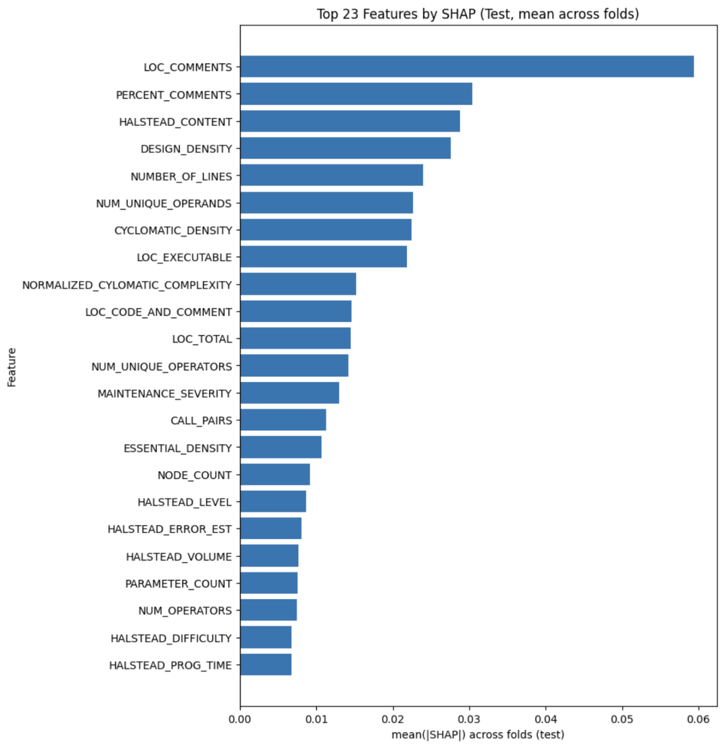

Figure 2: Top features by mean absolute SHAP on test (averaged across folds).

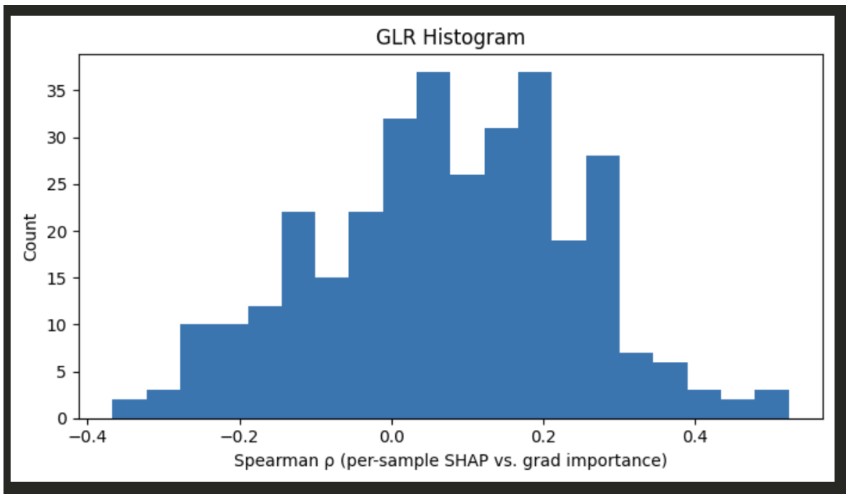

Figure 3: GLR histogram: per-sample Spearman ρ between SHAP and normalized sensitivity.

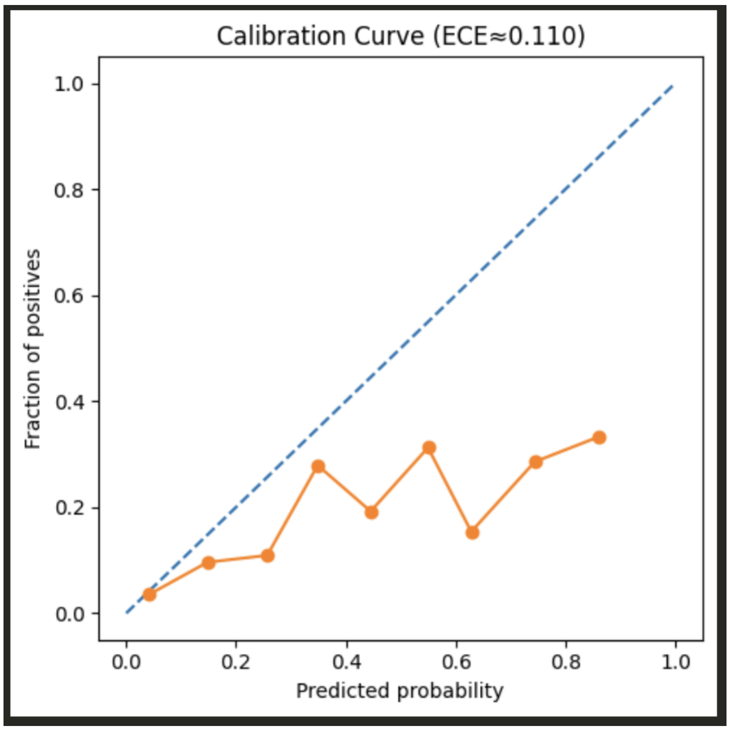

Figure 4: Calibration curve with ECE annotation.

7.11 Hyperparameters & Defaults (as in code)

- $\varepsilon = 10^{-3}$, `top_k` $= 10$, `m_limit` $= 20$, `calib_bins` $= 10$.
- `max_eps_samples` $= 50$ (global budget for ε-Hit@k sampling).
- `n_splits` $= 5$, `rng` $= 42$.

8 Dataset and Problem Setting

We use the **NASA PROMISE CM1** software defect dataset (`cm1.csv`). CM1 corresponds to a NASA spacecraft instrument (C code) and is widely studied for **module-level defect prediction**. Each row is a software module described by static code metrics; the binary target is `Defective` (1 = defective, 0 = non-defective).

Size and label.

- **Instances (modules):** 327
- **Attributes (features):** 35 numeric metrics (listed below)
- **Target (class):** `Defective` (1 or 0); in our file: 1 = 42, 0 = 285

Attribute list (35 features).

1. `BRANCH_COUNT` — number of branches in control flow.
2. `CALL_PAIRS` — number of function call pairs.
3. `LOC_CODE_AND_COMMENT` — lines containing both code and comments.
4. `LOC_COMMENTS` — comment-only lines.

5. `CONDITION_COUNT` — count of boolean conditions.
6. `CYCLOMATIC_COMPLEXITY` — McCabe cyclomatic complexity.
7. `CYCLOMATIC_DENSITY` — normalized cyclomatic complexity (complexity density).
8. `DECISION_COUNT` — number of decision points (e.g., `if`, `case`).
9. `DESIGN_COMPLEXITY` — design-level McCabe complexity.
10. `DESIGN_DENSITY` — normalized design complexity (density).
11. `EDGE_COUNT` — number of edges in the control-flow graph.
12. `ESSENTIAL_COMPLEXITY` — essential (structuredness) complexity.
13. `ESSENTIAL_DENSITY` — normalized essential complexity (density).
14. `LOC_EXECUTABLE` — executable (code) lines.
15. `PARAMETER_COUNT` — number of function/method parameters.
16. `HALSTEAD_CONTENT` — Halstead content metric.
17. `HALSTEAD_DIFFICULTY` — Halstead difficulty.
18. `HALSTEAD_EFFORT` — Halstead effort.
19. `HALSTEAD_ERROR_EST` — Halstead estimated errors (bugs).
20. `HALSTEAD_LENGTH` — Halstead length.
21. `HALSTEAD_LEVEL` — Halstead level (inverse difficulty).
22. `HALSTEAD_PROG_TIME` — Halstead programming time estimate.
23. `HALSTEAD_VOLUME` — Halstead volume.
24. `MAINTENANCE_SEVERITY` — maintenance severity indicator.
25. `MODIFIED_CONDITION_COUNT` — count of modified conditions (MC/DC-related).
26. `MULTIPLE_CONDITION_COUNT` — count of multi-part conditions.
27. `NODE_COUNT` — number of nodes in the control-flow graph.
28. `NORMALIZED_CYCLOMATIC_COMPLEXITY` — scaled cyclomatic complexity.
29. `NUM_OPERANDS` — total operands (Halstead).
30. `NUM_OPERATORS` — total operators (Halstead).
31. `NUM_UNIQUE_OPERANDS` — unique operands (Halstead).
32. `NUM_UNIQUE_OPERATORS` — unique operators (Halstead).
33. `NUMBER_OF_LINES` — total number of lines.
34. `PERCENT_COMMENTS` — percentage of comment lines.
35. `LOC_TOTAL` — total lines of code (LOC).

These metrics jointly capture size/volume (e.g., LOC, Halstead length/volume), structural and control-flow properties (e.g., nodes/edges/branches), and different notions of complexity (cyclomatic, essential, design), all of which are known to correlate with defect proneness.

NASA PROMISE CM1 Dataset (cm1.csv)

9 Model and Training Controls

Model. We use a **Random Forest** classifier (tree ensemble) rather than an MLP. Key hyperparameters (as used in code):

```
n_estimators=600,
min_samples_leaf=2,
max_depth=None,
class_weight="balanced_subsample",
n_jobs=-1.
```

Each fold uses a fixed `random_state` (fold id).

Training control. Random Forests do not use learning-rate schedules or early stopping. Regularization is controlled by ensemble size and tree shape (`min_samples_leaf`, `max_depth`). We report standard generalization metrics per fold (AUC, F1, Precision, Recall, Brier).

Class imbalance. We address imbalance via two mechanisms aligned with the code:
(i) **SMOTE** oversampling is applied on *training folds only* to synthesize minority-class samples;
(ii) **class weighting** with `balanced_subsample`, which re-weights classes within each bootstrap sample. (No `pos_weight` or logits-based BCE are used for training, since RF is nonparametric.)

Explainability setup. We compute per-sample SHAP attributions on test data using **TreeExplainer** with an interventional background of size $\min(256, |X_{\text{train}}|)$ and `model_output` = `probability`. Global importance is the mean absolute SHAP over test samples.

BCE on Predicted Probability (used in reliability checks)

Although the Random Forest is not trained with BCE, we use the **binary cross-entropy on probabilities** $\ell(p, y)$ as a smooth, model-agnostic loss to probe reliability:

$$\ell(p, y) \;=\; -\Big[y \log p + (1-y)\log(1-p)\Big], \quad p = \Pr(y = 1 \mid x).$$

In code, p is clipped to $[10^{-8},\, 1 - 10^{-8}]$ for numerical stability.

True-Loss Sensitivity via ε-Perturbations

For a sample $x \in \mathbb{R}^d$ and feature j, we estimate the magnitude of the loss sensitivity by a central finite difference:

$$g_j \;\approx\; \left|\frac{\partial \ell}{\partial x_j}\right| \;\approx\; \frac{\big|\ell\big(p(x+\varepsilon e_j), y\big) - \ell\big(p(x-\varepsilon e_j), y\big)\big|}{2\varepsilon},$$

with small $\varepsilon = 10^{-3}$. We then normalize per sample,

$$\tilde{g}_j \;=\; \frac{g_j}{\sum_{k=1}^{d} g_k + \epsilon},$$

to obtain a comparable importance profile used to (i) compute **GLR** (Spearman rank alignment between SHAP and $\tilde{g}$) and (ii) validate explanations via **ε-Hit@k** (do SHAP top-k features also yield the largest $\Delta\ell$ under $+\varepsilon$ nudges?).

Calibration and Reliability Aggregation

Calibration quality is summarized by the **Expected Calibration Error** (ECE) computed over uniform probability bins. We also report a composite **ReliabilityScore** that averages a rescaled GLR, ε-Hit@k, and $(1-\text{ECE})$ (clipped at 0), providing a single, interpretable summary of explanation alignment, perturbation validity, and probability honesty.

10 Evaluation: Discrimination, Thresholding, Calibration

Discrimination (AUC). We compute the **ROC AUC** on test-fold probabilities from the Random Forest and report the *mean across folds*. AUC is threshold-free and summarizes ranking quality of positives vs. negatives.

Thresholding (0.5) & Class Metrics. Hard predictions are obtained via a fixed threshold $\hat{y} = \mathbf{1}\{\, p \geq 0.5 \,\}$. We report **F1**, **Precision**, and **Recall** per fold on the test split, then average across folds. (Accuracy and Youden's J are *not* used in this pipeline.)

Calibration: Brier score & ECE. We assess probability quality with the **Brier score** and the **Expected Calibration Error** (ECE), using uniform probability bins. Let p_i be the predicted probability for sample i and $y_i \in \{0, 1\}$ the true label.

$$\text{Brier} \;=\; \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2.$$

Let the predictions be partitioned into B bins; in bin b, $\text{conf}(b)$ is the mean predicted probability and $\text{acc}(b)$ the empirical positive rate. Then

$$\text{ECE} \;=\; \sum_{b=1}^{B} \frac{n_b}{N} \,\big|\, \text{acc}(b) - \text{conf}(b) \,\big|.$$

We also form *reliability diagrams* via `calibration_curve` (uniform bins), plotting fraction of positives vs. mean predicted probability, and annotate the curve with the computed ECE.

Fold Aggregation. Per-fold metrics (AUC, F1, Precision, Recall, Brier) are computed on each test fold and averaged. For calibration plots and ECE, all test-set probabilities and labels from all folds are concatenated before binning to obtain a single overall reliability curve.

11 Explanation Methods

11.1 SHAP with Interventional TreeExplainer (as used in code)

We compute SHAP values on the test split with **TreeExplainer** (Random Forest):

- **Background:** a random training-fold sample of size $\min(256, |X_{\text{train}}|)$ (no K-means), with `feature_perturbation=interventional`.
- **Model output:** `probability`. For binary tasks, we take the positive-class SHAPs and use $|\phi|$.
- **Global importance:** mean of $|\phi|$ across test samples; we visualize top features by $\text{mean}|\phi|$.

11.2 True-Loss Sensitivity via ε-Perturbations (Finite Differences)

Setup. Let $x \in \mathbb{R}^d$ be a test sample, $y \in \{0, 1\}$ its label, and $p(x) = \Pr(y = 1 \mid x)$ the Random Forest probability. We use the **binary cross-entropy** on probability

$$\ell(p, y) \;=\; -\big(y \log p + (1 - y) \log(1 - p)\big),$$

with p clipped to $[10^{-8}, 1 - 10^{-8}]$ for stability (as in code).

Finite-difference sensitivity per feature j. For a small $\varepsilon > 0$ (code uses $\varepsilon = 10^{-3}$),

$$g_j \;\approx\; \left|\frac{\partial \ell}{\partial x_j}\right| \;\approx\; \frac{\big|\ell\big(p(x + \varepsilon e_j), y\big) - \ell\big(p(x - \varepsilon e_j), y\big)\big|}{2\varepsilon},$$

where e_j is the j-th unit vector.

Per-sample normalization.

$$\tilde{g}_j \;=\; \frac{g_j}{\sum_{k=1}^{d} g_k + \epsilon},$$

yielding a distribution $\tilde{g}$ over feature importance for the sample.

Tiny Illustrative Example

For a 2-feature sample $x = (1.00, 2.00)$, $y = 1$, $\varepsilon = 10^{-3}$. Suppose the RF gives:

$$p(x_1{=}1.001, x_2{=}2.00) = 0.82, \quad p(x_1{=}0.999, x_2{=}2.00) = 0.78.$$

Then $\ell^+ = -\log(0.82) \approx 0.198$, $\ell^- = -\log(0.78) \approx 0.248$,

$$g_1 \approx \frac{|0.198 - 0.248|}{0.002} \approx 25.0.$$

Similarly, if $p(x_1{=}1.00, x_2{=}2.001) = 0.81$, $p(x_1{=}1.00, x_2{=}1.999) = 0.805$, $\ell^+ = 0.2107$, $\ell^- = 0.2168$,

$$g_2 \approx \frac{|0.2107 - 0.2168|}{0.002} \approx 3.05.$$

Normalize: $(\tilde{g}_1, \tilde{g}_2) \approx (0.891,\ 0.109)$.

12 Reliability Criteria and Metrics

12.1 Gradient–Loss Reliability (GLR)

Let $\mathbf{s}_i \in \mathbb{R}^d$ be the vector of *absolute* SHAP values for sample i, and let $\tilde{\mathbf{g}}_i \in \mathbb{R}^d$ be the *normalized* finite-difference sensitivities (true loss sensitivity) for the same sample. We compute the per-sample **Spearman** rank correlation

$$\rho_i \;=\; \text{Spearman}(\mathbf{s}_i, \tilde{\mathbf{g}}_i)\,,$$

and report the **GLR** score as the average across n samples,

$$\text{GLR} \;=\; \frac{1}{n}\sum_{i=1}^{n} \rho_i \;\in [-1, 1].$$

Higher is better: it indicates that SHAP's ranking agrees with how the model's loss locally reacts to feature nudges.

What is Spearman correlation? Spearman correlation (Spearman's ρ) is a *rank correlation*: it measures agreement in *ordering*, not in numerical scale. This is ideal for feature importance because SHAP values and gradient/sensitivity magnitudes can live on different scales; if explanations are reliable, the *order* of important features should still match.

Formula

Given vectors

$$u = (u_1, \ldots, u_d), \qquad v = (v_1, \ldots, v_d),$$

first convert each to ranks:

$$r_j^u = \mathrm{rank}(u_j), \qquad r_j^v = \mathrm{rank}(v_j).$$

Then compute the **Pearson correlation of the ranks**:

$$\rho \;=\; \frac{\sum_{j=1}^{d}\big(r_j^u - \bar{r}^u\big)\big(r_j^v - \bar{r}^v\big)}{\sqrt{\sum_{j=1}^{d}\big(r_j^u - \bar{r}^u\big)^2}\sqrt{\sum_{j=1}^{d}\big(r_j^v - \bar{r}^v\big)^2}}.$$

Closed form without ties:

$$\rho \;=\; 1 - \frac{6\sum_{j=1}^{d}(r_j^u - r_j^v)^2}{d(d^2-1)}.$$

Interpretation

- $\rho \approx 1$: almost identical rankings $\Rightarrow$ explanations consistent with loss sensitivity.
- $\rho \approx 0$: no rank agreement $\Rightarrow$ explanations do not reflect sensitivity.
- $\rho < 0$: opposite rankings $\Rightarrow$ explanations contradict sensitivity.

Thus, **GLR** summarizes explanation reliability as the mean Spearman ρ over samples.

12.2 ε-Hit@k (Perturbation Validity)

Definition (plain language). ε-Hit@k tests whether the features that **SHAP** says are most important for a prediction are *actually* the ones that most increase the model's loss when we make a tiny change.

- "ε" = a tiny feature nudge (e.g., +0.001).
- "Hit" = at least one of SHAP's top-k features also appears among the features causing the largest loss increases after nudging.
- "@k" = how many top features we compare (top-1, top-3, top-5, ...).

If the two top-k sets overlap, the sample is a hit (1); otherwise 0. Averaging over samples yields the ε-Hit@k score.

Step-by-step formulation. For a single sample $x \in \mathbb{R}^d$ with label y:

1. Let T_k be the indices of the **top-k** features ranked by $|\phi|$ (SHAP). In code we first restrict to the top $M = \min(\texttt{m\_limit}, d)$ SHAP features and take $T_k \subseteq \text{Top-}M$.
2. For each candidate feature $j \in \text{Top-}M$, perturb one-at-a-time:
$$x^{+j} \;=\; x + \varepsilon e_j, \qquad \Delta\ell_j \;=\; \ell\big(p(x^{+j}), y\big) \;-\; \ell\big(p(x), y\big),$$
where $p(\cdot)$ is the model probability and ℓ is BCE on probability.
3. Let D_k be the **top-k** features ranked by $\Delta\ell_j$.
4. Define the per-sample indicator:
$$\text{Hit@k(sample)} \;=\; \mathbf{1}\{\, T_k \cap D_k \neq \emptyset \,\}.$$

Finally, over the evaluated samples,

$$\varepsilon\text{-Hit@k} \;=\; \frac{\#\text{hits}}{\#\text{tested samples}}.$$

Pipeline (as implemented).

1. In each fold, randomly select $\approx$ 15–20% of test samples, respecting a **global budget** `max_eps_samples`.
2. For each chosen sample: compute SHAP $\Rightarrow T_k$; for each of the top-M SHAP features, add $+\varepsilon$ and compute $\Delta\ell_j \Rightarrow D_k$; record hit $= \mathbf{1}\{T_k \cap D_k \neq \emptyset\}$.
3. Average the hits across all chosen samples and folds to report ε-Hit@k.

Simple numeric example. One sample, three features (A, B, C), true label $y = 1$, $k = 2$, $\varepsilon = 0.001$, $M \geq 3$.

- **SHAP (absolute) rankings:**
$$|\phi_A| = 0.30, \quad |\phi_B| = 0.20, \quad |\phi_C| = 0.05 \;\Rightarrow\; \text{Top-}M = [A, B, C],\; T_2 = \{A, B\}.$$
- **Loss after $+\varepsilon$ perturbations:** (baseline loss = 0.20)
$$\begin{aligned} A &: \ell(x{+}\varepsilon e_A) = 0.30 \Rightarrow \Delta\ell_A = +0.10, \\ B &: \ell(x{+}\varepsilon e_B) = 0.25 \Rightarrow \Delta\ell_B = +0.05, \\ C &: \ell(x{+}\varepsilon e_C) = 0.205 \Rightarrow \Delta\ell_C = +0.005. \end{aligned}$$
Thus the $\Delta\ell$ ranking is $A > B > C \Rightarrow D_2 = \{A, B\}$.
- **Compare:** $T_2 = \{A, B\}$ and $D_2 = \{A, B\}$ intersect $\Rightarrow$ Hit@2(sample) $= 1$.

If instead $D_2 = \{C, B\}$, the intersection is $\{B\} \Rightarrow$ still a hit; but if $D_2 = \{C\}$, then no overlap $\Rightarrow$ hit $= 0$.

12.3 Calibration: Expected Calibration Error (ECE)

What it measures. The **Expected Calibration Error (ECE)** summarizes how well predicted probabilities match actual outcomes. A *well-calibrated* model that says "70% chance of defect" will be correct about 70% of the time for such predictions; an *over/under-confident* model will systematically over/under-shoot these frequencies.

How it is computed. Partition predicted probabilities into B uniform bins. In bin b, let $\mathrm{conf}(b)$ be the mean predicted probability and $\mathrm{acc}(b)$ the empirical positive rate. Then

$$\mathrm{ECE} \;=\; \sum_{b=1}^{B} \frac{n_b}{N} \left|\mathrm{acc}(b) - \mathrm{conf}(b)\right|,$$

where N is the total number of samples and n_b is the number falling in bin b. Small $\mathrm{ECE} \approx 0$ indicates excellent calibration; larger values indicate worse calibration.

Simple numeric example. Consider 10 predictions and labels:

Sample	1	2	3	4	5	6	7	8	9	10
$\hat{p}$	0.10	0.15	0.20	0.30	0.40	0.55	0.60	0.70	0.80	0.90
y	0	0	0	1	0	1	1	1	1	1

Using $B = 5$ bins: $[0, 0.2), [0.2, 0.4), [0.4, 0.6), [0.6, 0.8), [0.8, 1.0]$, we obtain:

$$\begin{aligned}
&\text{Bin 1: } n_1 = 2,\ \mathrm{conf}(1) = 0.125,\ \mathrm{acc}(1) = 0 && \Rightarrow \frac{n_1}{N}\left|\mathrm{acc} - \mathrm{conf}\right| = \tfrac{2}{10} \cdot |0 - 0.125| = 0.025,\\
&\text{Bin 2: } n_2 = 3,\ \mathrm{conf}(2) = 0.300,\ \mathrm{acc}(2) = \tfrac{1}{3} \approx 0.333 && \Rightarrow \tfrac{3}{10} \cdot |0.333 - 0.300| = 0.0099,\\
&\text{Bin 3: } n_3 = 1,\ \mathrm{conf}(3) = 0.55,\ \mathrm{acc}(3) = 1 && \Rightarrow \tfrac{1}{10} \cdot |1 - 0.55| = 0.045,\\
&\text{Bin 4: } n_4 = 2,\ \mathrm{conf}(4) = 0.65,\ \mathrm{acc}(4) = 1 && \Rightarrow \tfrac{2}{10} \cdot |1 - 0.65| = 0.070,\\
&\text{Bin 5: } n_5 = 2,\ \mathrm{conf}(5) = 0.85,\ \mathrm{acc}(5) = 1 && \Rightarrow \tfrac{2}{10} \cdot |1 - 0.85| = 0.030.
\end{aligned}$$

Summing the contributions gives

$$\mathrm{ECE} \;\approx\; 0.025 + 0.0099 + 0.045 + 0.070 + 0.030 \;=\; 0.18,$$

i.e., an 18% calibration error for this toy example. If $\mathrm{acc}(b) = \mathrm{conf}(b)$ in every bin, then $\mathrm{ECE} = 0$ (perfect calibration).

12.4 Composite Reliability Score (reported in code)

We rescale GLR to $[0, 1]$ via $\rho' = (\rho + 1)/2$ and report

$$\mathrm{ReliabilityScore} \;=\; \mathrm{mean}\Big(\rho',\ \varepsilon\text{-Hit@k},\ \max\{0,\ 1 - \mathrm{ECE}\}\Big).$$

13 Experimental Setup

Cross-validation. We use **5-fold Stratified** CV with a fixed random seed (`rng`=42). Within each fold, SMOTE is applied *only to the training split* (adaptive k-neighbors) to address class imbalance, and a **Random Forest** (600 trees) is trained. Per-fold we compute test metrics (AUC, F1, Precision, Recall, Brier) and aggregate their means across folds.

Artifacts (as produced by the code). The pipeline returns:

- `per_fold_metrics`: table of fold-wise AUC/F1/Precision/Recall/Brier.
- `glr_rhos`: vector of per-sample Spearman ρ (SHAP vs. normalized loss sensitivity).
- `all_proba`, `all_y`: concatenated test probabilities and labels for calibration/ECE.
- `top_shap_mean_test`: mean absolute SHAP per feature (averaged across folds).

Background for SHAP. SHAP values on the test split are computed with **TreeExplainer** using an *interventional* background formed by a **random sample** of the training fold of size $\min(256, |X_{\text{train}}|)$ (*not* K-means). Model output is set to `probability` and absolute SHAP values are used for importance.

True-loss sensitivity & GLR. For each test sample and feature, we estimate $|\partial\ell/\partial x_j|$ via central finite differences on the BCE loss with a small perturbation $\varepsilon = 10^{-3}$; sensitivities are row-normalized to $\tilde{g}$. **GLR** is the per-sample Spearman rank correlation between absolute SHAP and $\tilde{g}$, averaged over all test samples.

ε-Hit@k sampling. To keep perturbation testing efficient, in each fold we select a $\sim$ 15–20% random subset of test samples, subject to a **global budget** `max_eps_samples` $= 50$. For each selected sample, we take the top-M SHAP features ($M = \min(\texttt{m\_limit}, d)$), nudge each by $+\varepsilon$, compute $\Delta\ell$, and record a hit if SHAP top-k overlaps the $\Delta\ell$ top-k. The reported **ε-Hit@k** is the mean hit rate over sampled rows.

Calibration. Using all concatenated test probabilities/labels, we compute **ECE** with uniform bins (`calib_bins` $= 10$) and plot a reliability diagram via `calibration_curve`. A composite **ReliabilityScore** averages a rescaled GLR, ε-Hit@k, and $(1 - \text{ECE})$ (clipped at 0) for a single summary measure.

14 Results

From the current run (summary row):

Discrimination

- **AUC (mean across folds)**: 0.7315
- **F1 (mean)**: 0.2349
- **Precision (mean)**: 0.2982
- **Recall (mean)**: 0.2444
- **Brier (mean)**: 0.1322

Reliability of Explanations & Calibration

- **GLR (mean Spearman ρ)**: 0.0742
- **ε-Hit@10**: 1.0
- **ECE**: 0.1098
- **Composite ReliabilityScore**: 0.8091
- **eps_samples_used**: 50 (global perturbation budget reached)

Notes. AUC indicates moderate ranking performance; F1/Recall are low, consistent with class imbalance. ECE ≈ 0.11 suggests reasonably calibrated probabilities. GLR is small (~ 0.07), implying weak overall rank alignment between SHAP and true-loss sensitivity on average, whereas Hit@10 is maximal (likely aided by a large $k = 10$ relative to M), yielding a high composite ReliabilityScore.