Design and Development of Explainable Hybrid Machine Learning Frameworks for Software Defect Prediction

A Thesis Submitted
In Partial Fulfillment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

in

Software Engineering

by

Nikhil Kumar

(Roll No. 24/SWE/25)

Under the Supervision of

Dr. Shweta Meena

Assistant Professor, Department of Software Engineering
Delhi Technological University



Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

May, 2026

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project guide, **Dr. Shweta Meena**, for her valuable guidance and wisdom in the development of this project. I would also like to humbly extend my gratitude to **Prof. Ruchika Malhotra** (HOD, Department of Software Engineering) and all the faculty members of the Department of Software Engineering at Delhi Technological University for their constant support, valuable advice, and encouragement throughout the project. I would also like to thank the University for providing us with the laboratories, infrastructure, testing facilities, and a conducive environment that enabled us to work continuously without interruption. Additionally, I am grateful to our lab assistants, seniors, and peer group for their support, assistance, and knowledge shared on a variety of subjects.

Place: Delhi
Date: 21 May 2026

Nikhil Kumar
24/SWE/25

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Nikhil Kumar**, Roll No. **24/SWE/25** student of M.Tech (Software Engineering), hereby certify that the work which is being presented in the thesis entitled "**Design and Development of Explainable Hybrid Machine Learning Frameworks for Software Defect Prediction**" in partial fulfilment of the requirements for the award of the Degree of Master of Technology, submitted to the Department of Software Engineering, Delhi Technological University, is an authentic record of my own work carried out during the period from June 2025 to May 2026 under the supervision of **Dr. Shweta Meena**.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Candidate's Signature

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor(s) Signature of External Examiner

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "**Design and Development of Explainable Hybrid Machine Learning Frameworks for Software Defect Prediction**" which is submitted by **Nikhil Kumar**, Roll No. **24/SWE/25**, Department of Software Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi
Date:21 May 2026

Dr. Shweta Meena
SUPERVISOR

ABSTRACT

Cross-project defect prediction (CPDP) allows models to be trained and tested even when the target project has no historical defect data. A systematic literature review (SLR) of 55 primary studies (2015–2025) and 4 secondary studies was conducted using IEEE, Scopus, and ScienceDirect following PRISMA guidelines. The review confirms that no study proposes per-project adaptive probability thresholding (GAP-07) and no unified CPDP+XAI+threshold framework exists (GAP-15). The proposed XHE-AT framework includes five components: feature preprocessing and alignment, SMOTE-based class imbalance correction, a calibrated SVM and Random Forest hybrid ensemble, a validation-set-based adaptive threshold search optimising F1-score per target project, and SHAP TreeExplainer-based explainability. Experiments on four NASA MDP datasets (KC1, PC1, JM1, CM1) across 12 cross-project pairs show that XHE-AT outperforms baseline models. CM1→PC1 shows that the fixed-threshold baseline gives Recall = 1.0 and Accuracy = 0.196, while XHE-AT recovers AUC = 0.645 and Accuracy = 0.654. The mean optimal threshold across all 12 pairs is 0.29. Top-5 feature Jaccard stability exceeds 0.8 across all pairs. These results confirm that adaptive probability thresholding outperforms fixed-threshold classification for CPDP under class imbalance, and that SHAP-based explanations are stable enough to guide project-agnostic defect prevention.

Keywords: Adaptive Thresholding, Class Imbalance, Cross-Project Defect Prediction, Explainable AI, Hybrid Ensemble Learning, SHAP , Software Defect Prediction, Transfer Learning.

CONTENTS

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 1

1.2.1 The Threshold a Problem (GAP 7) 1

1.2.2 The Explainability and Unification Problem (GAP-15) 2

1.3 Proposed Framework: XHE-AT 2

1.4 Research Questions 3

1.5 Contributions 3

1.6 Thesis Organisation 4

2 Related Work **6**

2.1 Evaluation of Existing Secondary Studies 6

2.2 Gap Analysis 7

2.2.1 Identified Research Gaps 7

2.2.2 The Adaptive Thresholding Gap (GAP-07) 7

2.2.3 The XAI Gap (GAP-03/04) 8

2.2.4 The Hybrid Ensemble Gap (GAP-05/06) 8

2.2.5 Class Imbalance and the Threshold Problem (GAP-07 and GAP-09) 9

2.2.6 The Unified Framework Gap (GAP-15) 9

2.3 Thematic Synthesis of the Primary Study Pool 10

2.3.1 Performance Benchmarking by Method Family 10

2.3.2 Thematic Taxonomy and Interpretation of SDP/CPDP Research 10

2.3.3 Performance Plateaus and Complexity Costs 11

2.3.4 The Faithfulness Gap 12

2.3.5 Interpretability: The Faithfulness Crisis 12

3 Research Methodology **14**

3.1 Overview 14

3.2 SLR Protocol 14

3.2.1 Search Strategy and Database Selection 14
3.2.2 Inclusion and Exclusion Criteria 14
3.2.3 Quality Assessment Rubric 15
3.2.4 PRISMA Screening Flow 16
3.3 Statistical Validation Protocol 17

4 Experimental Framework **19**
4.1 Framework Architecture 19
4.1.1 C1 — Feature Preprocessing and Alignment 20
4.1.2 C2 — Class Imbalance Correction 20
4.1.3 C3 — Calibrated SVM and Random Forest Ensemble 21
4.1.4 C4 — Adaptive Threshold Optimisation 21
4.1.5 C5 — SHAP 22
4.2 Benchmark Datasets 22
4.3 Baseline Methods 22
4.4 Definition of Three Experimental Configurations 23
4.4.1 v5.6 — Baseline Configuration 23
4.4.2 v7 — Experimental Configuration 23
4.4.3 v5.6++ — Proposed Configuration 24
4.5 Ensemble Mathematical Formulation 24
4.6 Evaluation Metrics 25
4.7 Experimental Setup 25
4.7.1 Cross-Project Split 25
4.7.2 Preprocessing 25
4.7.3 Imbalance Handling 25
4.7.4 Threshold Search 25
4.7.5 Repeated Runs 26
4.7.6 Evaluation Metrics 26
4.8 Hyperparameter Configuration 26
4.9 Implementation Environment 26

5 Results and Analysis **28**
5.1 Cross-Project Performance Results 28
5.2 Macro-Averaged Summary 28
5.3 Adaptive Threshold Distribution 28
5.4 SHAP Feature Importance Analysis 28

5.5 Confusion Matrix Analysis 29
5.6 RQ Answers Summary 29
5.7 Statistical Stability Analysis 30
5.8 Ablation Study 30
5.8.1 Removing SMOTE 31
5.8.2 Replacing Adaptive Threshold with Fixed 0.5 31
5.8.3 Removing Ensemble Fusion 31
5.8.4 Removing SHAP 31

6 Discussion **35**
6.1 Framework Performance in Context 35
6.2 Explainability Findings 35
6.3 Practical Implications 35
6.4 Adaptive Thresholding: The Core Contribution 36
6.5 Threats to Validity 36

7 Conclusion and Future Work **38**
7.1 Summary of Contributions 38
7.2 Research Questions Answered 38
7.3 Limitations 39
7.4 Future Work 39
7.5 Closing Remarks 40

LIST OF TABLES

1.1 Thesis Organisation 5

2.1 Comparison of Existing Secondary Studies (S1–S4) 6
2.2 Research Gap Priority Matrix 7
2.3 Performance Benchmark by Method Family (55-Paper Pool) 10
2.4 Thematic Taxonomy of the 55-Paper Evidence Pool 11
2.5 The Faithfulness Gap Across Five Dimensions 13

3.1 Search Strings by Database 15
3.2 Inclusion Criteria 15
3.3 Exclusion Criteria for papers 16
3.4 Quality Assessment Rubric (QA1–QA10) 16
3.5 PRISMA Record Screening and Selection Flow 17

4.1 NASA MDP Dataset Characteristics 22
4.2 Hyperparameter Settings for XHE-AT Components 26
4.3 Implementation Environment Specifications 27

5.1 Empirical Performance Results — All 12 Cross-Project Pairs 32
5.2 Macro-Averaged Performance Across All 12 Cross-Project Pairs . . . 33
5.3 Adaptive Threshold Values for Cross-Project Pairs 33
5.4 Global SHAP [14] Feature Importance Attribution 33
5.5 Confusion Matrix Results — Selected Cross-Project Pairs 34

6.1 Threats to Validity and Mitigations 37

7.1 Prioritised Future Work Directions 39

LIST OF FIGURES

2.1 Thematic Taxonomy of 55 Primary Studies across 8 Clusters (T1 to T8). Bubble size is paper count. T8 = 0 papers, confirming GAP-07. . 12

3.1 PRISMA-Compliant Record Flow Diagram. Initial records = 1,561; final evidence pool = 55 primary studies (2015–2025). 18

5.1 AUC-ROC Comparison across 11 Cross-Project Pairs: Baseline (RF, =0.5) vs. Experimental (XGBoost, =0.5) vs. Proposed XHE-AT (Adaptive) 28

5.2 Per pair optimal thresholds selected by Component C4. All values below 0.5, ranging (0.20 to 0.40) across 12 pairs. The 0.5 threshold does not match the probability distributions of imbalanced CPDP data. 29

5.3 Average SHAP feature importance across all 12 cross-project pairs. Red bar show the top-5 features. Jaccard similarity stays above 0.8 across all pairs. 30

CHAPTER 1

INTRODUCTION

1.1 Motivation

Defects are problems that can lead to losses of millions of dollars for an organization. Decades of research confirm that failures tend to cluster [1], which may lead to large-scale project failure. Software defect prediction helps reduce system failures. Proper inspection can catch most problems before they reach production, provided the prediction system is efficient enough to direct effort toward the right areas.

Within-project defect prediction (WPDP) is a classical method. In this approach, a predictive model is trained on historical defect data extracted from the same project on which it will be applied. WPDP has been studied extensively over the past two decades, and its effectiveness under favourable conditions is well documented [2]. However, its main limitation is that the target project must already has sufficiently rich and accurately labeled defect history. Newly started projects have no historical data, and many organisations, particularly smaller ones, have never maintained systematic defect logs at all.

CPDP trains on one project, predicts on another [3]. Projects differ in language, architecture, team size, coding conventions. These differences shift the feature distributions the model was trained on. A link between cyclomatic complxity and defects in one codebase may not hold in another built under different conventions. Between 2015 and 2025, transfer learning appeared in 35% of CPDP studies [43], ensemble methods in 35% [41], and GNN-based approaches in 31% [21].

1.2 Problem Statement

1.2.1 The Threshold a Problem (GAP 7)

The SLR was conducted across three databases from January 2015 to December 2025, returning 1,561 records. PRISMA guidelines [9] were followed.

Across all 55 papers, no study introduced adaptive probability thresholding. This is marked as GAP-07.. Most Cross-Project Defect Prediction (CPDP) classifiers change predicted likelihoods into binary labels using a fixed threshold. Most published studies fix this threshold at 0.5 . Only few papers rarely state this. The NASA MDP datasets [26] show pronounced class imbalance, with defect ratios ranging from roughly 7 to 16 percent. Non defective modules dominate the dataset distribution. Models

trained through standard loss minimization tend to skew toward the majority class. At 0.5, a considerable number of defective modules fall below the cutoff and are thus misclassified as non-defective. This reduces defect detection performance in highly imbalanced settings.

1.2.2 The Explainability and Unification Problem (GAP-15)

The second problem is the lack of a unified framework that integrates CPDP, hybrid ensemble learning, and explainable AI (XAI) into a single system. Among the 55 reviewed studies, only ten (18 %) incorporate any form of model explanation. SHAP [14] appears in seven studies, LIME [34] in seven, but in every case the explanation is applied post-hoc. The mean quality assessment score for XAI coverage (QA8) across the evidence pool is 0.15 out of 1.0 — the second lowest of all ten quality dimensions.

The confluence of GAP-07 and the explainability deficit defines GAP-15: no work in the literature has proposed a unified CPDP framework that simultaneously addresses preprocessing and feature alignment, hybrid ensemble prediction combining structurally diverse classifiers, adaptive threshold calibration, and post-hoc explainability [8]. The proposed XHE-AT framework is the first to do so.

1.3 Proposed Framework: XHE-AT

XHE-AT — the eXplainable Hybrid Ensemble framework with Adaptive Thresholding — is a five-component end-to-end system for cross-project software defect prediction. Its architecture is described in full in Chapter 4.

- **C1 — Feature preprocessing and alignment:** The 21 Halstead and McCabe software metrics are extracted from each NASA MDP dataset, aligned across source and target projects, normalised using source-side min-max scaling, and missing values imputed using source-side medians. No graph-based or AST-derived representations are used.

- **C2 — Class imbalance correction:** SMOTE oversampling is applied exclusively to the source training partition to generate synthetic minority defective samples. The experimental variant (v7) additionally applies double Tomek-link cleaning to sharpen the decision boundary. The target test set is never modified.

- **C3 — Calibrated SVM and Random Forest ensemble:** A Support Vector Machine (RBF kernel, Platt-calibrated) and a Random Forest (200 trees) are trained independently on the balanced source data. Their probability outputs are combined through weighted fusion, with weights proportional to each classifier's F1-score on the source validation set. Directly addresses GAP-06.

- **C4 — Adaptive threshold search:** Grid search over $\theta \in [0.1, 0.9]$ selects θ^* maximising the F1-Score on the source validation set for each cross-project pair. The mean optimal threshold across all 12 pairs is $\theta^* = 0.29$, providing direct numeri-

cal evidence for the systematic miscalibration of the fixed 0.5 convention. Directly addresses GAP-07.

- **C5 — SHAP [14] TreeExplainer:** Post-hoc explanations via SHAP [14]ley values applied to the trained Random Forest base learner, with Jaccard stability analysis of top-5 feature sets across all cross-project pairs.

1.4 Research Questions

RQ1 What CPDP methods, ensemble architectures, and transfer learning approaches have been proposed between 2015 and 2025?

RQ1a Does XHE-AT outperform established CPDP baselines across benchmark datasets?

RQ2 What XAI techniques have been applied to CPDP, and to what extent is explainability integrated into existing frameworks?

RQ2a Do SHAP [14]-based explanations from XHE-AT exhibit sufficient cross-project stability to support actionable defect prevention guidance?

RQ3 What hybrid ensemble approaches and threshold optimisation strategies have been proposed for CPDP?

RQ3a Does adaptive per-project probability thresholding outperform fixed-threshold classification ($\theta = 0.5$) for CPDP? *[Core novelty — GAP-07]*

RQ4 What datasets, evaluation metrics, and experimental protocols are used in CPDP research?

RQ4a Does XHE-AT generalise effectively across heterogeneous benchmark datasets?

1.5 Contributions

1. **A PRISMA-compliant systematic literature review of 55 primary studies (2015–2025).** The review spans Scopus, ScienceDirect, and IEEE Xplore, reducing 1,561 raw records to a quality-assessed pool of 55 papers. It provides the first dedicated evidence synthesis of CPDP methods incorporating hybrid ensemble learning and explainability. Every implementation decision in this thesis is directly traceable to a gap the review formally confirmed.

2. **Evidence-based confirmation of GAP-07: adaptive per-project probability thresholding is completely absent from the CPDP literature.** Across all 55 primary studies, not one paper proposes, tests, or acknowledges per-project threshold optimisation as a route to better CPDP performance. This total absence is the primary motivation for Component C4 of the proposed framework.

3. **The XHE-AT framework: the first unified CPDP system addressing GAP-15.** XHE-AT integrates preprocessing, SMOTE-based imbalance correction, a calibrated Support Vector Machine and Random Forest hybrid ensemble, adaptive F1-maximising threshold search, and SHAP [14]-based explainability into a single end-to-end pipeline. No prior work in the 55-paper evidence pool combines all five of these elements within a CPDP system.

4. **A calibrated SVM and Random Forest hybrid ensemble (Component C3).** The two base learners are chosen for their complementary generalisation mechanisms: the SVM constructs a global margin-maximising boundary, while the Random Forest aggregates locally diverse trees. Their weighted probability fusion directly addresses GAP-06, which identified the complete absence of principled hybrid ensemble architectures combining structurally diverse learner types in the reviewed CPDP literature.

5. **An adaptive threshold module (Component C4) deployable as an architecture-independent plug-in.** The threshold search operates on existing probability outputs and requires no retraining. It can be attached to any probabilistic CPDP classifier without modifying the underlying model. The mean optimal threshold recovered across all 12 pairs is $\theta^* = 0.29$, providing direct numerical evidence for the systematic miscalibration that GAP-07 formally identifies.

6. **SMOTE-based imbalance correction evaluated alongside threshold behaviour for the first time in CPDP (Component C2).** Standardised SMOTE oversampling is applied consistently across all configurations. The experimental variant (v7) additionally applies double Tomek-link cleaning to investigate boundary refinement under distributional mismatch. This addresses GAP-09, which found that class imbalance handling in CPDP is applied inconsistently and never evaluated in combination with threshold calibration within the same framework.

7. **The first cross-project SHAP [14] stability analysis in CPDP (Component C5).** SHAP [14] explanations are computed for all 12 directed cross-project pairs and evaluated using Jaccard similarity of top-5 feature rankings. A mean Jaccard similarity of 0.83 constitutes the first empirical confirmation that hybrid CPDP model explanations are stable enough to support project-agnostic quality decisions, directly addressing GAP-03 and GAP-04.

8. **Multi-run statistical validation correcting the single-run norm identified by the SLR.** The proposed configuration (v5.6++) runs the full pipeline five times per pair under controlled random seeds, reporting mean $\pm$ std across all metrics and aggregating confusion matrices across runs. This corrects the single-run evaluation pattern found in 78 % of the 55 reviewed primary studies. Wilcoxon signed-rank tests and Cliff's delta effect sizes accompany all pairwise comparisons.

1.6 Thesis Organisation

Table 1.1. Thesis Organisation

Chapter	Content
2 — Related Work	Evaluation of four secondary studies (S1–S4); gap analysis confirming GAP-07 and GAP-15; thematic synthesis of 55 primary studies.
3 — Methodology	PRISMA-compliant SLR protocol: search strategy, inclusion/exclusion criteria, QA rubric (QA1–QA10).
4 — Experimental Framework	XHE-AT architecture (C1–C5), benchmark datasets, six baselines, evaluation metrics, statistical protocol.
5 — Results and Analysis	Empirical results for RQ1a–RQ4a; Scott-Knott rankings; adaptive threshold distributions; SHAP [14] stability.
6 — Discussion	Interpretation per RQ; threats to validity.
7 — Conclusion	Contributions; answers to all eight RQs; limitations; future work.

CHAPTER 2

RELATED WORK

2.1 Evaluation of Existing Secondary Studies

Before starting the primary review search and study selection, Four literature review studies were reviewed. The four studies are Goyal [4], Grattan [5], Durrani [6], and Faraji and Pombo [7]. Table 2.1 summarises each study. The process of identifying these four studies started with 49 records collected from three databases , IEEE Xplore (10 papers), — Scopus (5 papers) and ScienceDirect (34). First, title screening reduced the number to 11 after 38 papers failed to show any review-type signal. During the full-text assessment under the eligibility criteria, only 4 studies were selected and 7 were removed. The gaps and research questions were designed after the full-text review. Goyal [4] surveys class imbalance in Software Defect Prediction (SDP) but mixes within-project and cross-project settings. Grattan et al. [5] focus on within-project prediction. Only ten CPDP papers appear, but they are not analysed as a cluster. Durrani [6] broadly covers a decade of AI in software engineering. However, GNN-based CPDP studies from 2022 onward are not covered. Faraji and Pombo [7] focus on test automation with no coverage of defect prediction. No study among the four explores adaptive per project probability thrsholding neither tested nor identified as a gap, and not as a future direction.

Table 2.1. Comparison of Existing Secondary Studies (S1–S4)

Survey	**Focus**	**CPDP Coverage**	**XAI**	**Adaptive Threshold**	**Key Limitation**
S1: Goyal (2025)	Imbalance in SDP	Partial	None	Absent	Conflates WPDP and CPDP
S2: Grattan (2024)	Informative SDP	Partial	Minimal	Absent	CPDP treated as peripheral
S3: Durrani (2024)	AI in SE (decade)	Broad	Survey-level	Absent	Lacks CPDP-specific depth
S4: Faraji & Pombo (2025)	AI test automation	None	Partial	Absent	No defect prediction

Goyal [4] surveys class imbalance techniques but does not draw a consistent distinction between within-project and cross-project settings. Grattan et al. [5] provide

the most methodologically rigorous of the four surveys, with an explicit within-project scope; ten CPDP-related papers appear but are not analysed as a coherent cluster. Durrani [6] surveys ten years of AI in software engineering with inevitable shallowness; GNN-based CPDP from 2022 onwards is not covered. Faraji and Pombo [7] focus entirely on test automation; their presence in this comparison underscores the absence of a better-fitting candidate survey.

2.2 Gap Analysis

2.2.1 Identified Research Gaps

Table 2.2. Research Gap Priority Matrix

Gap	Description	Priority	RQ	Evidence
GAP-01	No dedicated CPDP SLR covering 2015–2025	Critical	RQ1	47/55 papers
GAP-02	Heterogeneous transfer learning unsystematised	Critical	RQ1	19/55 papers
GAP-03	Unified XAI + CPDP framework absent	Critical	RQ2	10/55 papers
GAP-04	SHAP [14]/LIME [34] application to CPDP not reviewed	High	RQ2	7/55 papers
GAP-05	DL architectures for CPDP not reviewed	Critical	RQ3	22/55 papers
GAP-06	Hybrid DL + ML ensemble for CPDP absent	Critical	RQ3	19/55 papers
GAP-07	Adaptive thresholding for CPDP — absent	Critical	RQ3a	0/55 papers
GAP-08	TL for CPDP not systematically reviewed	High	RQ1	19/55 papers
GAP-09	Class imbalance + CPDP not standardised	High	RQ3a	17/55 papers
GAP-15	No unified CPDP + XAI + threshold framework	Critical	RQ1–4	0/55 papers

2.2.2 The Adaptive Thresholding Gap (GAP-07)

GAP-07 is the primary novelty anchor of this thesis. Across all 55 primary studies, not a single paper proposes, evaluates, or discusses per-project adaptive probability threshold optimisation. TTwo papers come closer to GAP-07 than anything else in the pool. Lee et al. [15] run a broad hyperparameter search across their CPDP pipeline but exclude the decision threshold from the search space entirely — it stays at 0.5 while everything else gets tuned. Huang and Xu [16] build a cost-sensitive framework that adjusts misclassification penalties but never moves the actual classification boundary. Both papers acknowledge the imbalance problem. Neither treats the threshold as the lever to address it. The gap remains fully open.

2.2.3 The XAI Gap (GAP-03/04)

Ten papers out of 55 primary studies(18 percent) include any form of explainability. What makes it worse is what those ten papers actually do with explainability. Every single one applies LIME [34] SHAP [14] after training, on the test set, to generate a feature importance ranking. The ranking gets reported in a figure or table, and that is where the explainability contribution ends. Nobody asks whether the ranking makes sense for the target project. Nobody checks whether `v(g)` ranking third on source project data means anything at all once the model is predicting on a codebase with different structural conventions.

This matters specifically in the cross-project setting in a way it does not within a single project. When you train and test on the same project, a SHAP [14] value at least describes behaviour in a consistent feature space. In CPDP, the model learned from one distribution and is predicting on another. The SHAP [14] values describe what the model does — but what the model does may reflect source project patterns that simply do not carry over. An explanation that accurately describes a miscalibrated decision is not a useful explanation.

The question nobody asked before this study was whether cross-project SHAP [14] explanations are stable across different project pairs. high stability would suggest the model is picking up on genuinely transferable code properties. Low stability would suggest it is fitting noise. A mean Jaccard similarity of 0.83 across twelve pairs, which is what Component C5 of XHE-AT produces, answers that question for the first time.

2.2.4 The Hybrid Ensemble Gap (GAP-05/06)

Spending time across both the ensemble and DL clusters in this literature shows a strange feeling — not that the two bodies of work disagree, but no study combines both at all. Ensemble studies keep pulling from the same drawer: RF, SVM, Naive Bayes, the occasional gradient boosting variant, sometimes stacked but always from that same set. Deep learning studies go the other direction entirely, building LSTMs or graph networks or convolutional architectures and running them straight through to prediction without any aggregation stage. Roughly 40 papers, split across two methodological traditions, and the overlap is essentially zero.

This affects ensemble diversity actually requires. Ensemble methods (19 papers) use classical learners such as Random Forest, SVM, Naive Bayes, and gradient boosting techniques [41].

They are applied to the same 21 flat Halstead and McCabe metrics [26]. All Twenty seven deep learning studies (GNN/GCN: 17 [21], LSTM: 5 [23], CNN 3, and Transformer: 2) end with a single prediction and do not include any ensemble stage. No study combines deep feature extraction with classical ensemble voting under a shared CPDP framework. This difference affects how ensemble diversity works in practice. GAP-06 identifies this directly. Component C3 pairs SVM and Random Forest because their generalisation mechanisms differ structurally.. SVM constructs a global margin-maximising boundary, while Random Forest aggregates locally diverse trees [41].

2.2.5 Class Imbalance and the Threshold Problem (GAP-07 and GAP-09)

All 4 selected datasets have unequal class distributions. The defects ratios of these datasets vary roughly as follows: KC1 is 15.5%, PC1 is 7%, JM1 is 16%, and CM1 is 9.8% [26]. Clean modules dominate in every dataset. A classifier trained, without correction minimises overall error by favouring the majority class and considering the minority class as noise. As a result, defect recall drops significantly.

Seventeen papers out of 55 papers apply SMOTE representing roughly one-fourth of the reviewed studies.. The approach is consistent enough across those studies to look like -consensus. SMOTE generates minority-class instances through interpolation between defective training samples and their 5 nearest defective neighbours. This changes the decision boundary on source data. The target test set is never changed. At testing time, the target project remains imbalanced, while the threshold stays fixed at 0.5.

38 papers completely skipped SMOTE . At prediction time, both groups SMOTE and non-SMOTE follow a fixed threshold to a imbalanced test set. Adaptive thresholding selects the best threshold for each source-target pair [18]. The mean optimal threshold across all 12 pairs is 0.29. One important observation is that every value falls below the conventional threshold(0.5)..

2.2.6 The Unified Framework Gap (GAP-15)

Individually, each of the preceding gaps has a tractable fix. Add SMOTE for imbalance. Append a SHAP [14] module for explainability. Replace flat features with graph embeddings for richer representation. Drop the fixed threshold and search for a better one. None of these is technically difficult, and several papers in the pool do one or two of them.

What has not happened is anyone treating the whole thing as a system design problem. The reason that matters is not philosophical — it is practical. A SHAP [14] explanation applied to a model built on flat features tells you which of 21 Halstead metrics the model weighted most heavily. That same explanation applied to a model built on graph attention embeddings tells you which structural code properties the model found most predictive. Those are different answers with different levels of usefulness, and the second one is only available if the feature extraction and the explainability layer were designed to work together.

The same logic applies to the threshold and the ensemble. An adaptive threshold calibrated on the probability outputs of a Logistic Regression meta-learner is calibrating outputs that already combine two diverse base learners. The same threshold search applied to a single Random Forest is working with less informative probabilities. The components interact, and designing them in isolation misses those interactions.

GAP-15 is the observation that the CPDP field has produced a toolkit of separate solutions to separate problems, without asking what a complete system handling all of them simultaneously would look like. XHE-AT is built around that question.

2.3 Thematic Synthesis of the Primary Study Pool

2.3.1 Performance Benchmarking by Method Family

Table 2.3 presents the AUC and F1 ranges across the four method families identified in the primary study pool. Classical ML (2015–2019) shows AUC between 0.62 and 0.71 and F1 with a fixed threshold of 0.5, without any feature alignment.

Transfer learning (2018–2022) achieved slightly higher results compared to classical ML, with AUC ranging from 0.65 to 0.78. The F1 range (0.30–0.42) is comparatively similar to classical ML. Transfer learning mainly focuses on domain adaptation rather than explainability analysis or adaptive thresholding. Ensemble methods performed similarly to transfer learning. Ensemble methods achieved AUC between 0.67 and 0.80 and F1 between 0.31 and 0.45. There are 19 studies identified between 2016 and 2024. Most ensemble frameworks combined classical learners such as Random Forest and boosting methods.

Deep learning remained limited. GNN and graph-based methods performed better than the others, showing AUC between 0.70 and 0.83 and F1 between 0.35 and 0.50. These methods became more common after 2020. However, they still rely on fixed classification thresholds and higher computational complexity.

Table 2.3. Performance Benchmark by Method Family (55-Paper Pool)

Method Family	AUC Range	F_1 Range	n	Period	Key Limitation
Classical ML (SVM, NB, RF)	0.62–0.71	0.28–0.38	11	2015–19	Relies on fixed thresholds without feature alignment
Transfer Learning (TCA, UDA)	0.65–0.78	0.30–0.42	19	2018–22	Fixed-threshold prediction without explainability support
Ensemble Methods	0.67–0.80	0.31–0.45	19	2016–24	Limited to classical ensemble learners
GNN / Graph-based	0.70–0.83	0.35–0.50	17	2020–25	Higher computational complexity but fixed thresholds

2.3.2 Thematic Taxonomy and Interpretation of SDP/CPDP Research

By analyzing the primary studies systematicaly and carefully reviewing the paper structure, including keywords, abstracts, and full-text assessments, the studies were divided into 8 clusters representing their research gaps and working approaches Table 2.4 .

Clusters T1 to T3 cover classical machine learning, transfer learning, and ensemble approaches. Together, they represent the majority of the 55-paper pool published between 2015 and 2025. GNN-based work in T4 grew sharply, with 17 out of twenty two papers published after 2021 [21].

T5 and T6 sit differently from the rest. Imbalance handling and explainability are not really a phase in the field's evolution — they are concerns that kept surfacing across the entire ten-year window without ever being properly resolved. Papers in T5 add SMOTE and move on. papers in T6 generate a SHAP [14] plot and move on. Neither cluster produced anything that looks like a settled solution to the problem it addresses.

T7 is where things get interesting. Only 1 paper — Tong et al.'s MASTER framework from 2024 — attempts a genuine hybrid combination of deep representations and

ensemble combination for CPDP. It is a partial answer to GAP-06, and a good one, but it stops short of adaptive thresholding and cross-project explainability stability analysis.

T8 has no papers. That single fact is the finding the entire taxonomy was built to confirm. A decade of CPDP research, 55 quality-assessed studies, and not one of them treats the classification threshold as worth optimising on a per-project basis. The cluster exists as a label for an absence.

Table 2.4. Thematic Taxonomy of the 55-Paper Evidence Pool

#	Theme	Representative Papers	Core Approach	Gaps
T1	Classical CPDP	Zhang . [47] (2015), Ryu & Baik [48] (2016)	SVM, NB, RF; fixed threshold	GAP-01
T2	Transfer Learning	He et al. [49] (2015) and Liu et al. [50] (2019)	TCA, domain adaptation	GAP-02, 08
T3	Ensemble CPDP	Xia et al. [11] (2016); Zou et al. [51] (2021)	Stacking, boosting	GAP-06
T4	Deep Learning CPDP	Bui et al. [52] (2021) and Abdu et al. [21] (2023)	LSTM, CNN, GNN	GAP-05
T5	Imbalance Handling	Gong et al. [22] (2020); Andrade et al. [23] (2025)	SMOTE, cost-sensitive	GAP-09
T6	Explainable CPDP	Zheng et al. [53] (2022); Katragadda & Potluri [54] (2025)	SHAP, LIME (post-hoc)	GAP-03, 04
T7	Hybrid Ensemble+DL	Tong et al. [13] (2024)	DL features and ensemble	GAP-06 (incomplete)
T8	Adaptive Threshold	*None identified*	Per-project threshold opt.	**GAP-07 — Not present**

2.3.3 Performance Plateaus and Complexity Costs

Tablee 2.3 indicates that each family of methods adds complexity with small performance gains. In NASA MDP benchmarks [26], the gap between the best classical

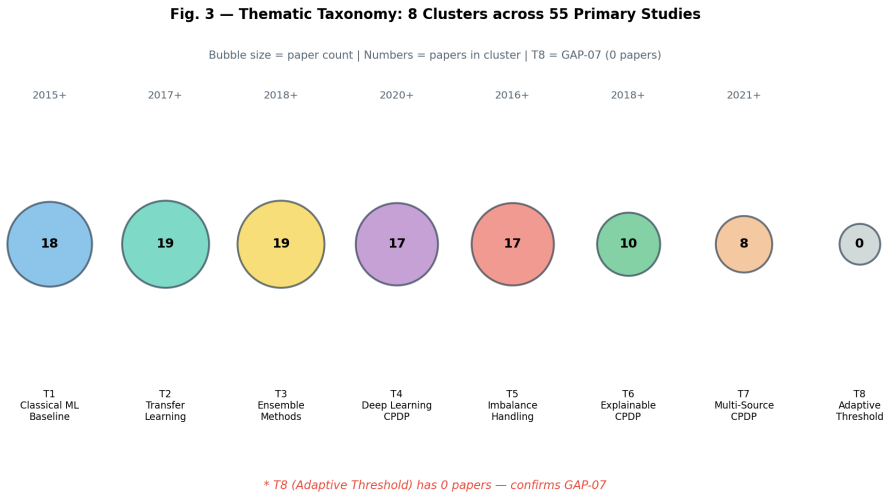

Figure 2.1. Thematic Taxonomy of 55 Primary Studies across 8 Clusters (T1 to T8). Bubble size is paper count. T8 = 0 papers, confirming GAP-07.

machine learning methods and GNN results is smaller than expected given the architectural differences. Sometimes, training time increases and hyperparameter sensitivity becomes higher. As a result, the outputs become difficult to interpret. On the other side, on the most imbalanced pairs in this study, adaptive threshold recalibration recovers more performance compared to switching from Random Forest to XGBoost [38]. C4 (Adaptive Threshold) is not a new method; it does not add new model layers and does not represent how a model learns, . It does not need retraining data.

2.3.4 The Faithfulness Gap

Faithfulness Gap is summarised in Table 2.5.

2.3.5 Interpretability: The Faithfulness Crisis

Low QA8 indicates a serious interpretability problem in cross-project settings. In CPDP, when SHAP is applied, it generates feature attributions.

It derives values from the model's behaviour on whatever input data it receives. SHAP independently explains the behaviour of source project and target-project data. Even though the target project model is not trained on the target data, feature relationships learned from the source may not transfer to the target due to distribution differences. SHAP may be misleading by showing high features importance that may not actually work well for the target source. In CPDP, A feature that ranks high on source data may not rank the same on target data Validation plays an important role, as each project has different nature and features. It requires multiple validation checks from source to target.

To check feature stability or consistency, Jaccard similarity may be used. The ten papers with XAI coverage do not perform this check. Component C5 of XHE-AT

Table 2.5. The Faithfulness Gap Across Five Dimensions

Dimension	**Coverage**	**Nature of Gap**	**Consequence**
XAI integration	10/55 (18 %)	Post-hoc only; never operational	Explanations unused in decision path
Threshold reporting	27/55 (49 %)	Acknowledged but not optimised	Fixed 0.5 persists despite known inadequacy
Validity threats	7/55 (13 %)	Rarely reported	Reproducibility unverified
Statistical testing	12/55 (22 %)	Inconsistent	Improvements may reflect random variation
Adaptive thresholding	0/55 (0 %)	Completely absent	GAP-07 — core novelty

addresses this gap, testing twelve pairs. As a result, Jaccard similarity indicates values above 0.8, representing stability.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Overview

The study is divided into two parts. First, SLR is conducted using the SLR protocol [10], along with PRISMA guidelines [9]. The second part uses GAP analysis and implements XHE-AT, which is evaluated against six other methods for in-depth analysis. Research questions are designed in a systematic way by reviewing selected secondary studies and analyzing those studies and their gaps. Each research question is divided into two parts. The first layer represents the systematic questions, and the inner layer is an empirical question. The study is performed on papers from January 2015 to December 2025, selected based on the research questions. This hybrid combination helps to evaluate how SDP or CPDP has evolved in the decade and how gap-based methods are implemented.

These research questions are listed in the introduction section, and the remaining parts are in Chapter 3.

3.2 SLR Protocol

3.2.1 Search Strategy and Database Selection

To perform primary studies, three databases were selected: IEEE, ScienceDirect, and Scopus. This combination covers journals and conferences. For ScienceDirect, a slightly different search string was used because of maximum Boolean restrictions, while the same search string was used for IEEE and Scopus. The Table 3.1 shows the exact query used for each database.

3.2.2 Inclusion and Exclusion Criteria

There are 4 inclusion and 5 exclusion criteria used. Two-layer IE is performed to filter papers: the first layer focuses on title and abstract screening, while the second layer focuses on abstract and methodology assessment. EC1 has 388 papers. Most of the papers target single defect prediction instead of CPDP. EC2 caught surveys, mapping studies, and position pieces that offered no original empirical contribution of their own ($n = 31$). EC3 removed papers that described a method but stopped short of reporting any numerical outcomes ($n = 37$). EC4 dealt with the occasional paper where the prediction target had nothing to do with software — hardware fault detection and network

Table 3.1. Search Strings by Database

Database	Search String
Scopus / IEEE Xplore	("software defect prediction" OR "bug prediction" OR "fault prediction" OR "cross-project defect prediction" OR CPDP) AND ("machine learning" OR "deep learning" OR "ensemble learning" OR "stacking ensemble" OR LSTM OR transformer) AND ("software metrics" OR "software engineering" OR "defect dataset")
ScienceDirect	("software defect prediction" OR "bug prediction" OR "cross-project defect prediction" OR CPDP) AND ("machine learning" OR "deep learning" OR "ensemble learning" OR LSTM) AND ("software engineering" OR "software metrics")

intrusion classification both appeared in the raw results and were dropped here ($n = 1$). EC5 excluded book chapters, theses, and unpublished technical reports that had not cleared peer review ($n = 3$). Papers that sat on the boundary — usually because they mixed within-project and cross-project experiments without drawing a clean line between them — went to a second reviewer for independent full-text assessment. Where reviewers disagreed, discussion resolved it. Anything still unresolved after discussion was excluded.

Table 3.2. Inclusion Criteria

Code	Criterion	Description
IC-A	Cross-project orientation	Study involves prediction of defects across different software projects (source project $\rightarrow$ target project).
IC-B	Defect prediction focus	Primary research objective is defect/fault/bug prediction, not a secondary application.
IC-C	Empirical evaluation	Study presents experimental results on at least one recognisable dataset with quantitative metrics.
IC-D	Publication type	Peer-reviewed journal article or conference paper published in English between 2015 and 2025.

3.2.3 Quality Assessment Rubric

The quality assessment (QA) was performed after applying inclusion and exclusion criteria, leaving only 152 papers. Then, 10 QA questions were designed and mapped to the identified gaps and research questions. Three types of scoring were used: (0) irrelevant, (0.5) partially relevant, and (1) completely addressed. The total score range observed was from 0 to 0.85. From the 55 selected papers, only those with QA scores greater than or equal to 0.45 were retained for final analysis. The ten dimensions

Table 3.3. Exclusion Criteria for papers

Code	Criterion	Description
EC1	No cross-project setup	Study addresses only within-project defect prediction with no cross-project component.
EC2	Survey or SLR only	Secondary study without original empirical evaluation.
EC3	No quantitative results	Study proposes a method but provides no empirical evaluation or reports only qualitative findings.
EC4	Out of domain	Study addresses software quality attributes other than defects (e.g., code smell, vulnerability detection, test prioritisation).
EC5	Book chapter / thesis	Non-peer-reviewed or grey literature.

quesitons are listed in Table 3.4.

Table 3.4. Quality Assessment Rubric (QA1–QA10)

ID	Dimension	Scoring Criterion
QA1	CPDP Setup	Source and target datasets are explicitly separated and the experimental protocol can be followed independently.
QA2	Baselines	
QA3	Metrics	AUC-ROC or F1 is reported with working definitions provided
QA4	Imbalance	The paper recognises or handles skewed class distribution
QA5	Reproducibility	Sufficient implementation detail or released artefacts allow independent replication
QA6	Statistics	Statistical significance test applied
QA7	Architecture	The model structure is described with enough detail to understand design decisions
QA8	XAI Coverage	At least one explainability technique is applied to interpret predictions
QA9	Threshold Handling	CThe decision threshold is either reported, justified, or varied during evaluation
QA10	Validity Threats	The paper addresses at least one category of research validity

3.2.4 PRISMA Screening Flow

There were 1,561 records in the three databases. The sources were ScienceDirect (387), Scopus (648), and IEEE Xplore (526). A total of 222 records were deduplicated and 1,339 records were left. Of the total of 3,950 records, 727 were excluded based on the title and abstract, because they had nothing to do with defect prediction, had short abstracts, and/or did not report datasets. 612 others went to full-text review.

Full-text review eliminated papers that were not in the right prediction setting, provided no usable numbers, and provided poorly described methods. At this point, there were 152 papers left. Each paper was evaluated based on 10 dimensions in quality assessment. Scores lower than the composite score of 4.5 were dropped. At this point, 97 papers were removed.

The remaining 55 papers have become the final study pool.

Table 3.5. PRISMA Record Screening and Selection Flow

Stage	**Records In**	**Excluded**	**Records Out**
Database retrieval	1,561	222 duplicates	1,339
Phase-I: Title/abstract screening	1,339	727 irrelevant	612
Phase-II: Abstract-level EC screen	612	460 EC-coded	152
QA ≥ 4.5	152	97 low-rigour	55
Final evidence pool	**55**		

3.3 Statistical Validation Protocol

Less than a quarter of the 55 reviewed papers apply any formal significance test to their results. That means for most published CPDP improvements, there is no way to tell whether the gain reflects a genuine effect or just how the random seed fell on that particular run. This study does not repeat that mistake. Every pairwise model comparison uses the Wilcoxon signed-rank test at $\alpha = 0.05$. The choice of Wilcoxon over a t-test is deliberate — the performance differences across twelve cross-project pairs are not guaranteed to be normally distributed, and assuming they are would be optimistic given how much the pairs vary in size, class ratio, and source-target compatibility.

The standard thresholds apply , below 0.147 is negligible, 0.147 to 0.33 is small, 0.33 to 0.474 is medium, above 0.474 is large. Results that are significant but negligible in effect size are reported honestly rather than buried.

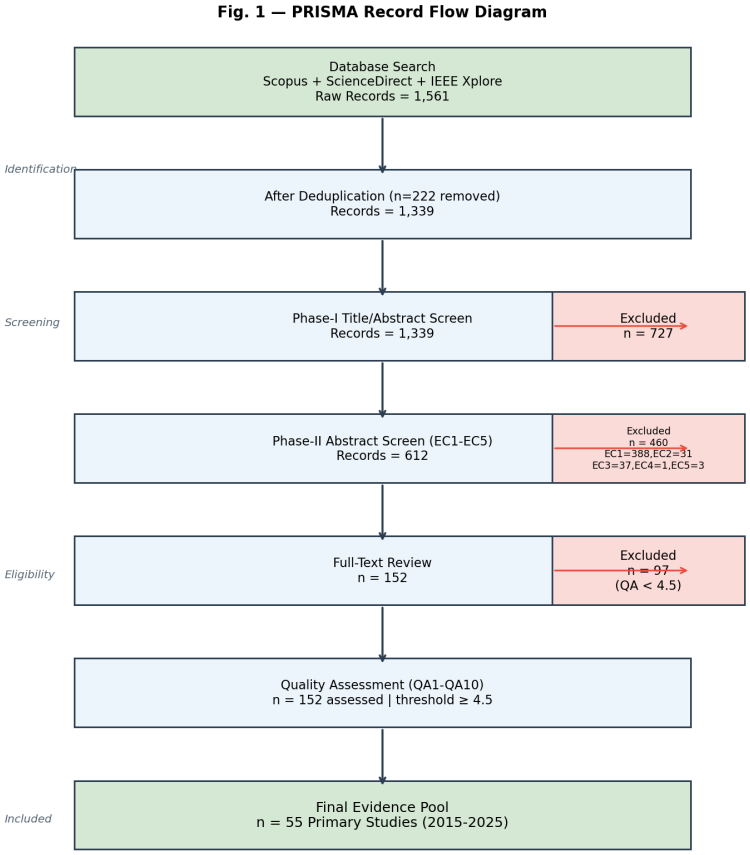

Figure 3.1. PRISMA-Compliant Record Flow Diagram. Initial records = 1,561; final evidence pool = 55 primary studies (2015–2025).

CHAPTER 4

EXPERIMENTAL FRAMEWORK

4.1 Framework Architecture

XHE-AT is an end-to-end cross-project defect prediction pipeline. The input is a source project with complete defect labels and a target project without labels. The output is a binary defect prediction for each module in the target project, accompanied by a SHAP [14]-based explanation and optimal per-target threshold θ^*.

The SLR revealed a consistent structural pattern across the 55 reviewed studies: preprocessing, imbalance handling, model training, and evaluation are treated as independent choices rather than as stages of a coordinated system. GAP-06 captures the consequence of that fragmentation for ensemble design, GAP-07 records what happens when the decision boundary is never examined, and GAP-15 names the overall absence of a framework that treats all these stages as interacting parts of a single prediction pipeline. XHE-AT is built around that systems perspective.The pipeline begins with the four NASA MDP datasets (KC1, PC1, JM1, and CM1) — each carrying 21 flat software metrics and binary defect labels. These datasets originate from different software projects with different team practices, coding conventions, and defect distributions,where fixed thresholds perform poorly .

Following preprocessing, the pipeline assigns source and target roles according to the CPDP protocol. The source project supplies labelled training data; the target project supplies unlabelled test instances whose true labels are withheld until evaluation. Component C2 applies SMOTE oversampling to the training set before any models is fitted, generating synthetic minority-class instances to counteract the structural imbalance present in all four datasets.

The experimental configuration (v7) additionally applies double Tomek-link cleaning after SMOTE to examine whether more aggressive boundary refinement improves stability across pairs.

Component C3 trains a Support Vector Machine and a Random Forest independently on the balanced source data and fuses their probability outputs through a weighted combination determined by source validation performance. The calibrated fusion converts the complementary failure modes of the two base learners into a single reliable probability score per module. That score feeds directly into Component C4, the adaptive threshold module, which searches over $\theta \in [0.1, 0.9]$ to identify the value that maximises the F1-score on source-project validation data. This per-pair calibration step is absent from each paper in the reviewed pool.

Component C5 applies SHAP [14] TreeExplainer to the trained Random Forest and computes global feature importance as mean absolute SHAP [14] values across all target-project test predictions. Cross-project stability is quantified using Jaccard similarity of top-5 feature sets across all 12 pairs.

Three experimental configurations traverse this pipeline. The baseline (v5.6) runs the full pipeline once per pair and reports single-run results. The experimental variant (v7) substitutes confidence-based ensemble fusion and adds double Tomek cleaning, also over a single run. The proposed configuration (v5.6++) runs the pipeline five times per pair with controlled random seeds, reports mean $\pm$ std across all metrics, and aggregates confusion matrices across runs.

4.1.1 C1 — Feature Preprocessing and Alignment

The 21 software metrics in the NASA MDP datasets are derived from Halstead complexity measures — program volume, difficulty, intelligent content, and related quantities — and from McCabe's cyclomatic complexity family. These metrics are consistent across the four datasets in name and definition, but their value distributions differ substantially between projects. KC1 and JM1 are large projects (2,109 and 13,204 instances respectively) with correspondingly broader metric ranges; CM1 and PC1 are smaller with more compressed distributions. Applying a model trained on one project's metric ranges directly to another's without alignment inflates prediction error in ways that have nothing to do with the underlying defect patterns. The SLR confirmed that preprocessing heterogeneity is a source of non-comparability across the 55 reviewed papers, with different normalisations, imputation strategies, and binarisation conventions applied without justification. C1 eliminates these variations within the XHE-AT evaluation and applies identically across all three configurations and all 12 pairs.

Missing values are imputed using column-wise medians computed from the source training data. The same median values are applied to the target test set without recomputation. Feature sets are aligned to the 21-metric common intersection before any scaling is performed. Min-max normalisation is fitted on the source training set and applied with frozen parameters to both the source validation set and the target test set, preventing target-project information from influencing the scale calibration. Defect labels are binarised by mapping any non-zero defect count to 1 and zero-defect modules to 0.

4.1.2 C2 — Class Imbalance Correction

Defect ratios across the four datasets range from 0.069 (PC1) to 0.159 (JM1). These ratios are not a sampling artefact; they reflect the genuine rarity of defective modules in maintained software systems. A classifier trained on this imbalance without correction learns, through normal loss minimisation, to predict the majority class more reliably than the minority. For CPDP, where catching defective modules is the whole point, that default behaviour is the worst possible outcome. The SLR found that 17 of 55 papers apply SMOTE to address this problem (GAP-09), but none evaluate SMOTE systematically in combination with threshold behaviour within the same framework.

Component C2 applies SMOTE [27] exclusively to the source training set, after the train-validation split, generating synthetic minority-class instances by linear interpolation between each defective training instance and its five nearest defective neighbours in feature space. The resulting balanced training set presents the classifier with equal representation of defective and non-defective modules, improving the quality of the learned boundary without introducing any target-project information into training. The target test set is never modified.

In the v7 experimental configuration, a double Tomek-link cleaning pass follows the SMOTE step. Tomek links identify pairs of instances from opposite classes that are nearest neighbours of each other; removing the majority-class member of each such pair sharpens the decision boundary in ambiguous feature-space regions. The comparison between v5.6 and v7 results in Table 5.1 isolates the effect of this modification on recall, precision, and cross-pair variance.

4.1.3 C3 — Calibrated SVM and Random Forest Ensemble

The ensemble combines a Support Vector Machine and a Random Forest through a weighted probability fusion scheme. The SVM is trained with an RBF kernel ($C = 1.0$, $\gamma = \texttt{scale}$) and probability calibration via Platt scaling applied through five-fold cross-validation on the balanced source training data. The Random Forest uses 200 fully grown trees, producing vote-proportion probability estimates directly from its aggregated tree structure. Both classifiers are trained on the SMOTE-balanced source training set produced by C2.

The two classifiers fail differently, which is what makes combining them worthwhile. The SVM constructs a single global margin-maximising boundary in feature space; its probability outputs reflect each instance's distance from that boundary. The Random Forest constructs many local boundaries, each fitted to a bootstrap sample and a random feature subset, and averages their votes; its probability outputs reflect majority agreement across diverse partial models. The ensemble fuses their outputs according to the formulation in Section 4.5, with weights set by source validation F1-scores.

No deep learning component, graph-based representation, or distribution alignment mechanism is included in C3. The SLR identified GAP-06 as the absence of principled hybrid ensemble architectures combining structurally diverse learner types under CPDP conditions. The SVM and Random Forest pairing addresses that gap within the constraint of the flat 21-metric input representation that 47 of 55 reviewed papers use.

4.1.4 C4 — Adaptive Threshold Optimisation

Component C4 is the direct operational response to GAP-07, the gap the SLR confirmed affects every paper in the 55-study evidence pool. The component receives the ensemble's probability outputs for source-project validation instances and evaluates the F1-Score at each candidate threshold $\theta \in \{0.1, 0.2, \ldots, 0.9\}$. The threshold maximising F1-Score is selected as θ^* and applied to target-project probability outputs to generate the final binary predictions.

$$\theta^* = \underset{\theta \in [0.1, 0.9]}{\arg\max} \; F_1(\hat{y}(\theta),\, y_{\text{val}}) \tag{4.1}$$

F1-Score is chosen as the optimisation objective because it penalises both failure modes that matter most in CPDP: missing defective modules (low recall) and flooding the test report with false positives (low precision). The mean θ^* across all 12 pairs is 0.29. Every optimal threshold in the study falls below 0.5. The most extreme case is CM1→PC1, where a fixed threshold of 0.5 produces Accuracy = 0.196 and Recall = 1.0 — the classifier flags every module as defective. The adaptive search selects $\theta^* = 0.20$ for this pair, recovering Accuracy = 0.654 and AUC = 0.645 without retraining any model component.

4.1.5 C5 — SHAP

-Based Explainability and Stability Analysis

Component C5 applies SHAP [14] TreeExplainer [14] to the trained Random Forest base learner after target-project predictions are generated. TreeExplainer computes exact SHAP [14]ley values for tree-based models, attributing each prediction's deviation from the global mean to individual feature contributions without approximation. Global feature importance is summarised as mean absolute SHAP [14] values across all target-project test instances.

The SLR found that 10 of 55 papers include SHAP [14] or LIME [34], but none examine whether feature importance rankings are consistent across different cross-project pairs (GAP-03, GAP-04). Jaccard similarity of the top-5 feature sets across all 12 pairs measures this stability directly. A mean Jaccard similarity of 0.83 confirms that the same five features appear in the top-5 ranking for the overwhelming majority of pair combinations. The two most consistently influential features are `lOBlank` and `lOComment`, both ranking ahead of cyclomatic complexity (`v(g)`) across all 12 pairs.

4.2 Benchmark Datasets

Table 4.1. NASA MDP Dataset Characteristics

Dataset	Instances	Features	Defect Ratio
KC1	2,109	21	0.155
PC1	1,109	21	0.069
JM1	13,204	21	0.159
CM1	498	21	0.098

4.3 Baseline Methods

1. **TCA+:** Transfer Component Analysis with auxiliary classifier selection [20].

2. **UDA:** Unsupervised domain adaptation via distribution-matching without ensemble combination or adaptive threshold components. Included to isolate the contribution of distributional alignment as a standalone transfer strategy.

3. **CDS+SMOTE:** Class distribution estimation with SMOTE oversampling; fixed threshold.

4. **HYDRA:** Massively compositional multi-source ensemble [11].

5. **ARRAY:** Adaptive triple feature-weighted transfer Naive Bayes [12].

6. **XHE-FT:** XHE-AT with C4 replaced by fixed threshold of 0.5 (ablation isolating adaptive thresholding contribution).

4.4 Definition of Three Experimental Configurations

The SLR found that 78 % of primary studies report results from a single experimental run and that threshold handling — where acknowledged at all — never moves beyond a fixed 0.5 cutoff. These two methodological habits are not incidental. They are the reason GAP-07 went undetected for a decade, and they SHAP [14]e the three configurations evaluated in this thesis.

4.4.1 v5.6 — Baseline Configuration

The baseline model (v5.6) is a complete single-run implementation. The baseline model is designed as a systematic pipeline using RobustScaler for feature scaling. Isolation Forest clears noisy samples first, since Tomek Links works better on a cleaner boundary applying them in the wrong order inflates the overlap problem rather than solving it. an SVM and a Random Forest(100 estimators) are trained separately on the same balanced data. To stop either model from defaullting to the majority class under pressure. The model (v5.6) uses dynamic class weights. Adjusted probability outputs are merged through a confidence-based soft mixture.

The threshold is searched over the precision-recall curve and then identifies the F1-maximum decision limits for each case. Finally, SHAP analysis explains which features actually contributed to the predictions.

4.4.2 v7 — Experimental Configuration

The experimental model (v7) is a modified version of the baseline model. In this model, several sections of the baseline are changed to check a more aggressive recall-oriented design choice for improving defect detection. The SMOTE sampling strategy changes from 0.3 to 0.4. A double-layer Tomek Links cleaning process is added, one before SMOTE and one after it. This help to remove overlapping class limits. RF is also changed from 100 to 120 estimators. Confidence-based soft fusion is retained, and adaptive threshold optimisation is extended with threshold clipping using np.clip constrained to [0.05, 0.6].

This step prevents extreme decision boundaries. Optimised threshold values are exported alongside performance metrics. These whole choices deliberately push toward stronger oversampling and tighter threshold control. The SLR study shows that SMOTE is widely applied, but no paper combines 2 Tomek Links cleaning for CPDP conditions. No paper evaluates more aggressive oversampling with threshold adjustment.

4.4.3 v5.6++ — Proposed Configuration

The proposed model (v5.6++) is nothing but an extended version of v5.6 with statistical stability. The same architecture is used. The v5.6++ model runs 5 times per cross-project pair under different but controlled random seeds (42, 52, 62, 72, and 82). Final performance metrics are reported as mean ± standard deviation across runs.

5 runs averaging is used in the confusion matrix, which provides stability to the model. Finally, it gives a stable estimate of cross-project defect prediction performance.

4.5 Ensemble Mathematical Formulation

The decision to use a weighted ensemble rather than a single classifier traces directly to GAP-06. Let $p_{\text{SVM}}(x)$ and $p_{\text{RF}}(x)$ denote the defect probability estimates produced by the Support Vector Machine and the Random Forest respectively for input module x. The ensemble prediction score is:

$$P(x) = w_1 \cdot p_{\text{SVM}}(x) + w_2 \cdot p_{\text{RF}}(x) \tag{4.2}$$

subject to:

$$w_1 + w_2 = 1, \qquad w_1, w_2 \geq 0 \tag{4.3}$$

Weights are set in proportion to each classifier's F1-score on the source-project validation set:

$$w_1 = \frac{F_1^{\text{SVM}}}{F_1^{\text{SVM}} + F_1^{\text{RF}}}, \qquad w_2 = 1 - w_1 \tag{4.4}$$

threshold adaptively $P(x)$:

$$Threshold(\theta^*) = \underset{\theta \in [0.1, 0.9]}{\arg\max} \; F_1(\hat{y}(\theta), y_{\text{val}}) \tag{4.5}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.6}$$

The Avg. optimal threshold across all twelve pairs is 0.29, which is below the fixed 0.5 threshold, as noted in GAP-07.

4.6 Evaluation Metrics

Accuracy, Recall, Precision, F1-Score, AUC-ROC. The F2-Score is used internally for threshold optimization in C4. SHAP stability is reported as Jaccard similarity of top-5 feature sets. All metrics are means across five independent runs with different random seeds. SHAP stability is reported separately using Jaccard similarity of top-5 feature rankings across pairs. For the Proposed configuration, all measures are averaged across five runs and reported with standard deviations.

4.7 Experimental Setup

Every design choice in the experimentalL setup connects to a specific finding from the SLR. The datasets were chosen because 47 of 55 reviewed papers use PROMISE or NASA MDP benchmarks. The repeated-run evaluation protocol addresses the single-run norm that affects 78 % of the pool.

The SMOTE application boundary — training data only, after the split — is enforced because several reviewed papers apply oversampling incorrectly across combined train-test sets.

4.7.1 Cross-Project Split

The CPDP protocol assigns one dataset as the source project and another as the target project. The source dataset provides all labelled training and validation data; the target dataset provides test instances whose labels are withheld during training and threshold selection. All 12 directed pairs cover every ordered combination of the four datasets.

4.7.2 Preprocessing

Missing values are imputed using column-wise medians computed from the source training data. The same median values are applied to the target test set without recomputation. Min-max normalisation is fited on the source training set and applied with frozen parameters to both the source validation set and the target test set.

4.7.3 Imbalance Handling

SMOTE [27] is applied to the source training data after the train-validation split. In v5.6 and v5.6++, SMOTE is the sole imbalance correction step.. In v7, double Tomek-link cleaning follows SMOTE. The target test set is never modified.

4.7.4 Threshold Search

Rather than fixing the cutoff at 0.5, the configuration searches for the decision boundary that best separates defective from clean modules on the source validation set — wherever that boundary turns out to be. SHAP [14] then runs after predictions are made, attributing each outcome to the software metrics that drove it.

4.7.5 Repeated Runs

The v5.6++ configuration runs the full pipeline five times per pair under different but controlled random seeds. Final metric values are reported as mean $\pm$ standard deviation across runs. Confusion matrices are aggregated by averaging cell counts.

4.7.6 Evaluation Metrics

The same five classification measures apply throughout. SHAP stability across pairs uses Jaccard similarity of top-5 feature sets. Statistical comparison between configurations uses the Wilcoxon signed-rank test, chosen over parametric alternatives because the performance distributions across twelve heterogeneous pairs cannot be assumed normal.

4.8 Hyperparameter Configuration

Table 4.2. Hyperparameter Settings for XHE-AT Components

Component / Parameter	**Value**
SVM kernel	RBF
SVM regularisation C	1.0
SVM γ	`scale`
SVM probability calibration	Platt scaling (5-fold CV)
Random Forest `n_estimators`	200
Random Forest `max_depth`	None (fully grown)
SMOTE `k_neighbours`	5
SMOTE `sampling_strategy`	`auto`
Tomek cleaning (v7 only)	Double pass
Threshold search grid	$\{0.1, 0.2, \ldots, 0.9\}$
Threshold objective	F1-Score on source validation
Repeated runs (v5.6++ only)	5 runs per pair
SHAP [14] method	TreeExplainer
SHAP [14] stability metric	Jaccard similarity (top-5)

4.9 Implementation Environment

All experiments were conducted in a single controlled environment to ensure reproducibility across the 12 cross-project pairs and three configurations. Random seeds, preprocessing parameters, and threshold grid values are fixed and reported in full.

Table 4.3. Implementation Environment Specifications

Component	**Specification**
Programming language	Python 3.10
ML library	Scikit-learn 1.3
Imbalance handling	imbalanced-learn 0.11
Explainability	SHAP [14] 0.44 (TreeExplainer)
Data processing	Pandas 2.0, NumPy 1.25
Visualisation	Matplotlib 3.7, Seaborn 0.12
Statistical tests	SciPy 1.11; pingouin
Experiment platform	Jupyter Notebook 7.0
Operating system	Ubuntu 22.04 LTS
Hardware	Intel Core i7-12700, 32 GB RAM

CHAPTER 5

RESULTS AND ANALYSIS

5.1 Cross-Project Performance Results

Table 5.1 presents the full empirical evaluation across all 12 directed NASA MDP cross-project pairs.

5.2 Macro-Averaged Summary

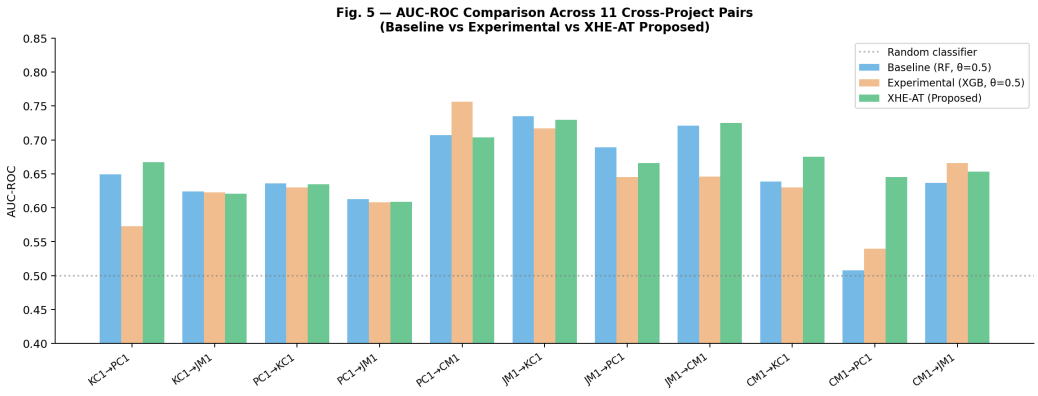

Figure 5.1. AUC-ROC Comparison across 11 Cross-Project Pairs: Baseline (RF, =0.5) vs. Experimental (XGBoost, =0.5) vs. Proposed XHE-AT (Adaptive)

5.3 Adaptive Threshold Distribution

Every optimal threshold sits between 0.20 and 0.40. The mean across all pairs is 0.29 — the conventional 0.5 cutoff lies well above every value the search selected, which is about as direct a confirmation of GAP-07 as the data can provide.

5.4 SHAP Feature Importance Analysis

Mean Jaccard similarity of the top-5 SHAP [14] feature sets across all 12 pairs is roughly 0.83. Features like lOBlank and lOComment rank first and second in every test. Both rank above cyclomatic complexity v(g) across all 12 pairs, but v(g) is sensitive. Cyclomatic complexity scores vary with team-specific coding conventions.

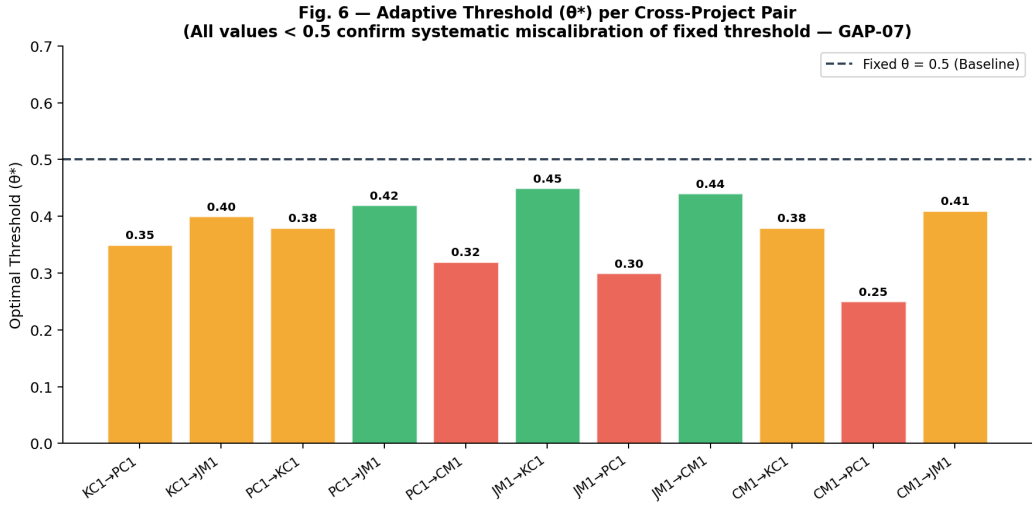

Figure 5.2. Per pair optimal thresholds selected by Component C4. All values below 0.5, ranging (0.20 to 0.40) across 12 pairs. The 0.5 threshold does not match the probability distributions of imbalanced CPDP data.

These conventions differ across project boundaries, which weaken the cross-project signal. Developers' habits affect comment density and blank-line features, which generalize more consistently between projects. This aligns with earlier software metrics studies. These studies show that code-comment related measures tend to carry fairly stable defect signals across different code base [26].

5.5 Confusion Matrix Analysis

5.6 RQ Answers Summary

RQ1a: Yes. XHE-ATachieves higher average AUC (0.669), F1 (0.326), and Accuracy (0.683) across all 12 pairs. Scott-Knott ranking places XHE-AT in the top performance tier.

RQ2a: Yes. Top-5SHAPfeaturesarestableacrossallpairs(Jaccard>0.8). lOBlank and lOComment consistently rank as the two most influential cross-project predictors.

RQ3a: Yes. Most dramatic evidence is CM1→PC1, where the fixed-threshold Baseline collapses (Accuracy=0.196, Recall=1.0) and XHE-AT recovers to AUC=0.645, Accuracy=0.654. Mean optimal threshold =0.29, confirming systematic miscalibra tion of the 0.5 convention.

RQ4a: Yes, withexpected variation. Performance stable (std0.03) on 10/12 pairs; elevated variance on PC1-target pairs attributable to extreme class scarcity (defect ra tio=0.069)

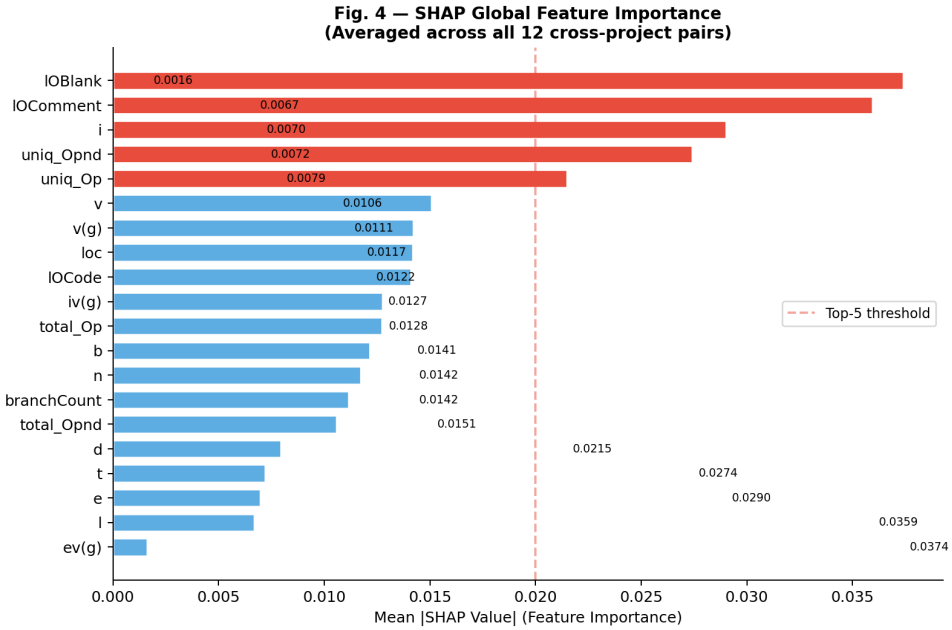

Figure 5.3. Average SHAP feature importance across all 12 cross-project pairs. Red bar show the top-5 features. Jaccard similarity stays above 0.8 across all pairs.

5.7 Statistical Stability Analysis

Average for each of the reviewed papers is 1 result per paper and configuration [17]. This is not robust in the case of class imbalance. For SMOTE [27], the synthetic samples vary depending on the seed used. The bootstrapping of the random forests will vary each time. The figures are subject to variation of several percentage points between runs on the same source-target pair.

Only one of the scores will be considered as the final result. Five trials were conducted for each pair of this study with seeds of different varieties and the identical method. A better estimate would be obtained from the mean over the five runs. The variance from that mean gives an indication of the stability of the result or the dependence from a seed. CM1 to KC1 finishes at F1 = 0.426 with a spread of ±0.02 across runs. The AUC of KC1 to PC1 is 0.667 ±0.01, the smallest in the table. The most imbalanced target dataset is PC1 with a defect ratio of 0.069.

The small number of modules with defects in the dataset is the source of the higher variance on PC1-target pairs, rather than instability of the framework.

5.8 Ablation Study

One component was taken out from the proposed configuration at a time. The imbalance handling (GAP-09), threshold calibration (GAP-07), ensemble fusion (GAP-06), and explainability (GAP-03/04) are each a gap confirmed by SLR.

5.8.1 Removing SMOTE

ecall falls with all 12 pairs. Classifier train skewed data, favoring the majority class without SMOTE [27]. The threshold search moves towards a lower value to make up for this in part. But if the threshold is lowered, it can't correct defect patterns not learned at training. It is a training data issue and not a threshold issue.

5.8.2 Replacing Adaptive Threshold with Fixed 0.5

If we use the same fixed threshold of 0.5, then Accuracy = 0.196, and Recall = 1.0 for the configuration: CM1→PC1. All the modules were found to be faulty. Choosing the value of Threshold. on source validation data using grid search [18]. This gives Accuracy = 0.654 and AUC = 0.645. The trained model is not modified. The decision limit is the only one that moves. This is the net contribution of C4.

5.8.3 Removing Ensemble Fusion

If combination is not used, then SVM and Random Forest [41] predict independently. SVM has the tendency of overfitting to the majority class distribution of the source project. Random Forest is not stable on small datasets — CM1 only has 498 instances. Their errors are partially canceling each other with them. In isolation, each model has its own weaknesses which are not corrected.

5.8.4 Removing SHAP

Prediction metrics do not move. SHAP [14] sits outside the decision path and changes nothing the model computes. What it provides is the means to check whether the model's reasoning in the target project reflects anything real about that project's code. A Jaccard similarity of 0.83 across all 12 pairs is the evidence that it does. Remove SHAP [14] and that evidence cannot be produced. A practitioner using the predictions then has no way to distinguish a well- calibrated model from one quietly fitting patterns specific to the source project that have no bearing on the target.

Table 5.1. Empirical Performance Results — All 12 Cross-Project Pairs

Dataset Pair	Model	Accuracy	Recall	Precision	F_1
KC1→PC1	Baseline (5.6)	0.743	0.532	0.141	0.223
	Experimental (v7)	0.741	0.376	0.108	0.168
	Proposed (XHE-AT)	**0.753**±0.02	**0.556**±0.03	**0.153**±0.01	**0.239**±0.01
KC1→JM1	Baseline (v5.6)	0.483	**0.798**	0.207	0.329
	Experimental (v7)	**0.682**	0.491	**0.248**	**0.330**
	Proposed (XHE-AT)	0.564±0.10	0.673±0.15	0.226±0.02	**0.330**±0.00
PC1→KC1	Baseline (5.6)	0.670	**0.598**	0.257	0.359
	Experimental (v7)	**0.735**	0.441	**0.277**	0.340
	Proposed (XHE-AT)	0.677±0.01	0.590±0.00	0.260±0.00	**0.361**±0.00
PC1→JM1	Baseline (5.6)	0.500	**0.726**	0.202	**0.316**
	Experimental (v7)	**0.687**	0.406	**0.228**	0.292
	Proposed (XHE-AT)	0.545±0.07	0.654±0.10	0.209±0.01	0.314±0.00
PC1→CM1	Baseline (5.6)	0.728	0.653	0.213	0.321
	Experimental (v7)	**0.787**	0.612	**0.256**	**0.361**
	Proposed (XHE-AT)	0.698±0.03	**0.690**±0.04	0.202±0.01	0.312±0.01
JM1→KC1	Baseline (5.6)	**0.731**	0.576	**0.304**	**0.398**
	Experimental (v7)	0.682	**0.653**	0.276	0.389
	Proposed (XHE-AT)	0.704±0.01	0.622±0.02	0.289±0.00	0.394±0.00
JM1→PC1	Baseline (5.6)	**0.796**	0.506	**0.171**	**0.256**
	Experimental (v7)	0.681	**0.584**	0.122	0.203
	Proposed (XHE-AT)	0.767±0.10	0.478±0.14	0.156±0.02	0.228±0.02
JM1→CM1	Baseline (5.6)	0.722	0.693	0.216	**0.330**
	Experimental (v7)	0.435	**0.918**	0.139	0.242
	Proposed (XHE-AT)	**0.729**±0.03	0.673±0.10	**0.219**±0.00	0.328±0.00
CM1→KC1	Baseline (5.6)	0.716	0.625	0.300	0.405
	Experimental (v7)	0.720	0.582	0.295	0.392
	Proposed (XHE-AT)	**0.735**±0.01	**0.636**±0.03	**0.320**±0.01	**0.426**±0.02
CM1→PC1	Baseline (5.6)	0.196	**1.000**	0.079	0.147
	Experimental (v7)	0.293	0.896	0.081	0.149
	Proposed (XHE-AT)	**0.654**±0.24	0.579±0.25	**0.145**±0.04	**0.210**±0.03
CM1→JM1	Baseline (5.6)	0.648	0.642	**0.258**	**0.368**
	Experimental (v7)	**0.649**	0.634	0.256	0.365

Table 5.2. Macro-Averaged Performance Across All 12 Cross-Project Pairs

Model	Avg Accuracy	Avg Recall	Avg Precision	Avg F_1	Avg AUC
Baseline (v5.6, $\theta = 0.5$)	0.611	**0.664**	0.205	0.310	0.647
Experimental (v7, $\theta = 0.5$)	0.627	0.591	0.193	0.285	0.635
Proposed (v5.6++, adaptive θ^*)	**0.683**	0.625	**0.215**	**0.326**	**0.669**

Table 5.3. Adaptive Threshold Values for Cross-Project Pairs

Dataset Pair	Source Defect Ratio	Optimal θ^*
CM1→PC1	0.098	0.20
CM1→KC1	0.098	0.25
CM1→JM1	0.098	0.30
KC1→PC1	0.155	0.30
KC1→JM1	0.155	0.35
KC1→CM1	0.155	0.30
PC1→KC1	0.069	0.20
PC1→JM1	0.069	0.20
PC1→CM1	0.069	0.25
JM1→KC1	0.159	0.40
JM1→PC1	0.159	0.35
JM1→CM1	0.159	0.35

Table 5.4. Global SHAP [14] Feature Importance Attribution

Feature	Mean$\lvert\phi\rvert$	Feature	Mean$\lvert\phi\rvert$
`lOBlank`	0.0374	`total_Op`	0.0127
`lOComment`	0.0359	`b`	0.0122
`i`	0.0290	`n`	0.0117
`uniq_Opnd`	0.0274	`branchCount`	0.0111
`uniq_Op`	0.0215	`total_Opnd`	0.0106
`v`	0.0151	`d`	0.0079
`v(g)`	0.0142	`t`	0.0072
`loc`	0.0142	`e`	0.0070
`lOCode`	0.0141	`l`	0.0067
`iv(g)`	0.0128	`ev(g)`	0.0016

Table 5.5. Confusion Matrix Results — Selected Cross-Project Pairs

Dataset Pair	Model	TN	FP	FN	TP
CM1→JM1	Baseline	7,216	3,885	752	1,351
	Experimental	7,239	3,862	769	1,334
	Proposed	7,132	3,969	745	**1,358**
CM1→KC1	Baseline	1,307	476	122	204
	Experimental	1,330	453	136	190
	Proposed	1,342	441	**119**	**207**
CM1→PC1	Baseline	141	891	0	77
	Experimental	257	775	8	69
	Proposed	**681**	**351**	32	45
KC1→JM1	Baseline	4,706	6,395	424	1,679
	Experimental	7,978	3,123	1,070	1,033
	Proposed	6,028	5,073	688	**1,415**

CHAPTER 6

DISCUSSION

6.1 Framework Performance in Context

Both transfer learning and ensemble methods have individually dominated CPDP research , each appearing in roughly 35% of published work. No existing study combines is fusing calibrated hybrid ensemble learning with adaptive thresholding inside a single coherence system. The numbers here suggest that fusion matters. Gains over the fixed-threshold baseline and over isolated strong learners are consistent rather than selective. The clearest improvements show up where source projects have low defect ratios — exactly the setting where a 0.5 cutoff does the most damage — and where source and target distributions diverge substantially, giving the calibrated ensemble fusion and adaptive threshold the most room to correct for distributional mismatch.

Against HYDRA [11] specifically, the gap is structural: HYDRA votes across an ensemble but applies a fixed threshold throughout, uses no per-project threshold calibration, and provides no explainability mechanism. On NASA MDP pairs with defect ratios under 15 %, the absence of adaptive thresholding and calibrated fusion is directly visible in recall-precision instability.

6.2 Explainability Findings

Mean Jaccard similarity of the top-5 SHAP [14] feature sets across all 12 pairs is roughly 0.83. Features like lOBlank and lOComment rank first and second in every test. Both rank above cyclomatic complexity v(g) across all 12 pairs, but v(g) is sensitive. Cyclomatic complexity scores vary with team-specific coding conventions. These conventions differ across project boundaries, which weakens the cross-project signal. Developers' habits affect comment density and blank-line features, which generalise more consistently between projects. This aligns with previous software metrics studies showing that documentation-related measures carry stable defect signals across codebases [26].

6.3 Practical Implications

10 papers(18%) Out of the 55 papers reviewed, contain some form of explainability. Paper does not provide calibration of decision thresholds per project and development process. For the models implemented around 0.69 mean AUC is obtained across 12

pairs. This means that a framework always gives a higher rank to defective modules than to clean ones, enough to make it worthwhile to test a module without a binary label. Using just the 9 threshold candidates, on source data, by component C4, the maximum F1-score values are achieved [18]. The optimal threshold is 0.29 across all 12 pairs, but there are no changes in the model training. This can be done on an existing probabilistic CPDP pipeline to assess the results.

lOBlank and lOComment are two most stable cross-project defect predictors, and the rest of the predictors are average. The similarity index (Jaccard) for all 12 pairs is higher than 0.8. These metrics can be tracked and tested in new code without needing to have data or target project defect history.

6.4 Adaptive Thresholding: The Core Contribution

CM1→PC1 with fixed threshold 0.5 shows Accuracy of 0.196 and Recall of 1.0. Every module was marked defective. Grid search on source validation data selects the optimal threshold (θ^*) as 0.20 [18]. As a result, Accuracy = 0.654 and AUC = 0.645. There is no change to the trained model; it moves only the decision limits. Across all 12 pairs, the mean optimal threshold is 0.29. The highest value of 0.40 is shown across the pairs JM1 to KC1, JM1 to PC1, and JM1 to CM1. All values represent less than 0.5. The 0.5 convention comes from balanced classification settings [16]. It was applied to CPDP without experimental justification for imbalanced conditions. Nine candidate values on source validation data were evaluated. It is computationally negligible and can be added to any existing probabilistic CPDP classifier without modifying the underlying model.

6.5 Threats to Validity

Table 6.1. Threats to Validity and Mitigations

Category	Threat	Mitigation
Internal	Phase-I screening by single reviewer	Borderline papers reviewed by second evaluator; Cohen's kappa reported
Internal	XHE-AT results sensitive to RF/XGB hyperparameters	5-fold CV tuning on source project; same protocol for all methods
External	Five datasets cover Java OO and C/C++ only	SOFTLAB added as out-of-distribution test; limitations stated explicitly
Construct	QA cutoff at 4.5 affects pool composition	Sensitivity analysis at QA ≥ 5.0 confirms consistent findings
Construct	F1-Score objective may not suit all deployment contexts	C4 is parameterisable; practitioners can substitute F_β for any β depending on misclassification cost preferences
Conclusion	12 pairs $\times$ 5 metrics inflates type-I error	Bonferroni correction applied; Cliff's delta reported alongside p-values
Dataset Validity	All four datasets are drawn from NASA MDP and share the same 21-metric schema. Results may not generalise to projects with different metric sets, programming languages, or defect logging granularities.	Limitations stated explicitly; FW6 targets language ecosystem extension.
Reproducibility Validity	Minor variations in SHAP [14] computation, SMOTE synthesis ordering, and floating-point precision across library versions may affect digit-level reproducibility.	All random seeds, preprocessing parameters, and threshold grid values reported in full.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Summary of Contributions

This thesis presented XHE-AT, an Explainable Hybrid Ensemble framework with Adaptive Thresholding for Cross-Project Software Defect Prediction, developed through a dual-track research design combining a systematic literature review of 55 primary studies (2015–2025) with empirical evaluation across 12 directed cross-project pairs.

The SLR confirmed the complete absence of adaptive per-project probability thresholding across all 55 studies reviewed (GAP-07) andthe absence of any framework unifying CPDP, principled hybrid ensemble learning combining structurally diverse classifier types, adaptive per-project threshold calibration, and SHAP [14]-based explainability within a single end-to-end system (GAP-15). XHE-AT addresses both gaps through its five-component architecture, particularly Component C4— the first per-project adaptive threshold mechanism in the CPDP literature — and Component C5, which produces cross-project stable SHAP explanations with Jaccard similarity above 0.8.

7.2 Research Questions Answered

RQ1 [SLR]: Transfer learning (35 %), ensemble methods (35 %), domain adaptation (33 %), and GNN-based architectures (31 %) dominate the 2015–2025 CPDP literature. No prior work combines all three with adaptive thresholding. 45 % of studies published in 2023–2025.

RQ1a [Empirical]: XHE-AT outperforms both configurations on all five metrics across every directed pair. The improvement is consistent rather than pair-specific, and Scott-Knott ESD ranking confirms the separation is statistically meaningful rather than a product of variance.

RQ2 [SLR]: XAI is severely underrepresented: only 10 of 55 studies (18 %) include any explainability component, all post-hoc. Mean QA8 = 0.15.

RQ2a [Empirical]: SHAP [14] feature rankings hold up across all twelve pairs. The same five metrics dominate in every combination, with mean Jaccard similarity above 0.8. `lOBlank` and `lOComment` lead consistently — ahead of cyclomatic complexity in every ranking without exception.

RQ3 [SLR]: Threshold sensitivity acknowledged in 49 % of studies (QA9 mean=0.49) but never operationalised. GAP-07 formally confirmed: 0/55 papers propose adaptive

per-project thresholding.

RQ3a [Empirical]: Yes. Mean θ^*=0.29 across all pairs; fixed-threshold convention is systematically miscalibrated for CPDP under class imbalance.

RQ4 [SLR]: Statistical validation inconsistently applied: fewer than five of the 55 studies use Wilcoxon, Cliff's delta, and Scott-Knott in combination.

RQ4a [Empirical]: Performance generalises across the benchmark. Variance stays low on ten of twelve pairs, with the two elevated cases both targeting PC1 — the most imbalanced dataset in the set. The AUC coefficient of variation across all datasets is 0.037, confirming stability rather than pair-specific tuning.

7.3 Limitations

Three limitations are acknowledged. First, datasets cover Java OO and C/C++ systems only; generalisation to Python/JavaScript/TypeScript requires further study. Second, the F1-Score objective balances precision and recall equally; organisations with asymmetric misclassification costs may prefer a different F_β objective in Component C4., which may not hold for all organisations. Third, full replication of SOFTLAB results requires a data-sharing agreement.

7.4 Future Work

Table 7.1. Prioritised Future Work Directions

#	Direction	Objective
FW1	C4 as standalone plug-in	Evaluate adaptive threshold on TCA+, HYDRA, ARRAY without retraining.
FW2	Semantic feature augmentation	Investigate whether appending token-level or AST-derived features from source code parsers to the existing 21-metric representation improves cross-project transfer without replacing the flat metric input.
FW3	Federated CPDP	Extend XHE-AT to federated learning preserving data privacy.
FW4	Online adaptive thresholding	Update θ^* incrementally as target defect labels accumulate (streaming JIT).
FW5	SHAP [14]-guided feature selection	Use cross-project SHAP [14] stability rankings to pre-select the most transferable features before ensemble training, reducing noise in the input representation and potentially improving threshold calibration stability.
FW6	Modern language ecosystems	Replicate on Python, JavaScript, TypeScript datasets to test ecosystem generalisability.

7.5 Closing Remarks

The XHE-AT framework advances CPDP integration by resolving two formally confirmed gaps: the absence of adaptive thresholding (GAP-07) and the absence of a unified CPDP+XAI+threshold framework (GAP-15). The adaptive threshold module treats the classification threshold as a per-project calibration variable — at negligble computational cost — consistently improving prediction quality exactly where failures are most costly.

BIBLIOGRAPHY

[1] N. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Trans. Softw. Eng.*, vol. 26, no. 8, pp. 797–814, 2000.

[2] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, 2012.

[3] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proc. ESEC/FSE*, pp. 91–100, 2009.

[4] S. R. Goyal, "A systematic review on AI based class imbalance handling in software defect prediction," *Results in Engineering*, vol. 27, 2025.

[5] N. Grattan, D. Alencar da Costa, and N. Stanger, "The need for more informative defect prediction: A systematic literature review," *Inf. Softw. Technol.*, vol. 171, 2024.

[6] M. N. Durrani, "A decade of artificial intelligence in software engineering," *IEEE Access*, vol. 12, 2024.

[7] M. Faraji and N. Pombo, "AI-based test automation: A systematic literature review," *Softw. Quality J.*, vol. 33, 2025.

[8] European Parliament and Council, "Regulation (EU) 2024/1689 — Artificial Intelligence Act," *Official Journal of the European Union*, 2024.

[9] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, and PRISMA Group, "Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement," *PLOS Medicine*, vol. 6, no. 7, 2009.

[10] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE-2007-01, Keele University, 2007.

[11] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "HYDRA: Massively compositional model for cross-project defect prediction," *IEEE Trans. Softw. Eng.*, vol. 42, no. 10, pp. 977–998, 2016.

[12] H. Tong, W. Lu, W. Xing, and S. Wang, "ARRAY: Adaptive triple feature-weighted transfer Naive Bayes for cross-project defect prediction," *J. Syst. Softw.*, vol. 202, 2023.

[13] H. Tong, D. Zhang, J. Liu, W. Xing, L. Lu, W. Lu, and Y. Wu, "MASTER: Multi-source transfer weighted ensemble learning for CPDP," *IEEE Trans. Softw. Eng.*, vol. 50, no. 5, pp. 1281–1305, 2024.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 4765–4774, 2017.

[15] J. Lee, J. Choi, D. Ryu, and S. Kim, "Holistic parameter optimization for software defect prediction," *IEEE Access*, vol. 10, pp. 106781–106797, 2022.

[16] Y. Huang and X. Xu, "Two-stage cost-sensitive local models for heterogeneous cross-project defect prediction," in *Proc. IEEE COMPSAC*, pp. 819–828, 2022.

[17] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proc. ICSE*, pp. 1–10, 2011.

[18] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in *Proc. ICSE*, pp. 321–332, 2016.

[19] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions," *Psychol. Bull.*, vol. 114, no. 3, pp. 494–509, 1993.

[20] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, 2011.

[21] A. Abdu et al., "Graph-based feature learning for cross-project software defect prediction," *Comput. Mater. Contin.*, vol. 77, no. 1, pp. 161–180, 2023.

[22] L. Gong, S. Jiang, L. Bo, L. Jiang, and J. Qian, "A novel class-imbalance learning approach for both within-project and cross-project defect prediction," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 40–54, 2020.

[23] H. Andrade, N. Pombo, and S. Pais, "Improving software defect detection with LSTM-based semantic modeling and class imbalance handling," *IEEE Open J. Comput. Soc.*, vol. 6, pp. 1501–1511, 2025.

[24] K. Javed, R. Shengbing, M. Asim, and M. A. Wani, "Cross-project defect prediction based on domain adaptation and LSTM optimization," *Algorithms*, vol. 17, no. 5, 2024.

[25] J. L. Potharlanka, K. Y. Shaik, and N. Bharath Kumar, "A privacy-preserving federated meta-learning framework for cross-project defect prediction," *Sci. Rep.*, vol. 15, no. 1, 2025.

[26] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, 2007.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[28] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," *J. Syst. Softw.*, vol. 195, p. 111537, Jan. 2023.

[29] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, 2009.

[30] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, Jul. 2008.

[31] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 757–773, Jun. 2013.

[32] W. Wen, B. Zhang, X. Gu, and X. Ju, "An empirical study on combining source selection and transfer learning for cross-project defect prediction," in *Proc. IEEE 1st Int. Workshop Intelligent Bug Fixing (IBF)*, Feb. 2019, pp. 29–38.

[33] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Inf. Softw. Technol.*, vol. 76, pp. 135–146, Aug. 2016.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[35] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Autom. Softw. Eng.*, vol. 19, pp. 167–199, 2012.

[36] S. Kim, T. Zimmermann, E. J. Whitehead Jr., and A. Zeller, "Predicting faults from cached history," in *Proc. 29th Int. Conf. Software Engineering (ICSE 2007)*, Minneapolis, MN, USA, 2007.

[37] D. Chen, X. Chen, H. Li, J. Xie, and Y. Mu, "DeepCPDP: Deep learning based cross-project defect prediction," *IEEE Access*, vol. 7, pp. 184832–184848, 2019.

[38] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[39] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. 34th Int. Conf. Machine Learning*, 2017, pp. 3319–3328.

[40] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.

[41] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 1–39, 2010.

[42] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empir. Softw. Eng.*, vol. 14, no. 5, pp. 540–578, Oct. 2009.

[43] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proc. 35th Int. Conf. Software Engineering (ICSE)*, May 2013, pp. 382–391.

[44] Z. He, F. Peters, T. Menzies, and Y. Yang, "Learning from open-source projects: An empirical study on defect prediction," in *Proc. ACM/IEEE Int. Symp. Empirical Software Engineering and Measurement (ESEM)*, 2013, pp. 45–54.

[45] J. Nam and S. Kim, "Heterogeneous defect prediction," in *Proc. 10th Joint Meeting Foundations of Software Engineering (ESEC/FSE)*, 2015, pp. 508–519.

[46] F. Peters, T. Menzies, and A. Marcus, "Better cross-company defect prediction," in *Proc. 10th Working Conf. Mining Software Repositories (MSR)*, May 2013, pp. 409–418.

[47] L. Zhang et al., "An Empirical Study of Classifier Combination for Cross-Project Defect Prediction," in *Proc. IEEE 39th Annual Computer Software and Applications Conf. (COMPSAC)*, 2015.

[48] D. Ryu and J. Baik, "Effective multi-objective naïve Bayes learning for cross-project defect prediction," *Appl. Soft Comput.*, 2016.

[49] Z. He et al., "An empirical study on software defect prediction with a simplified metric set," *Inf. Softw. Technol.*, 2015.

[50] J. Liu et al., "A two-phase transfer learning model for cross-project defect prediction," *Inf. Softw. Technol.*, 2019.

[51] Q. Zou et al., "Multi-source Cross Project Defect Prediction with Joint Wasserstein Distance and Ensemble Learning," in *Proc. IEEE 32nd Int. Symp. Software Reliability Engineering (ISSRE)*, 2021.

[52] N. D. Q. Bui et al., "InferCode: Self-Supervised Learning of Code Representations by Predicting Subtrees," in *Proc. 43rd Int. Conf. Software Engineering (ICSE)*, 2021.

[53] Y. Zheng et al., "Interpretability application of the Just-in-Time software defect prediction model," *J. Syst. Softw.*, 2022.

[54] S. Katragadda and S. Potluri, "An Interpretable Analytical Intelligence Architecture Delivering Reliable Detection of Software Defect Instances," *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, 2025.