SPECTRUM OF ENGINEERING SCIENCES
ISSN (E) 3007-3138 (P) 3007-312X VOLUME 4, ISSUE 3, 2026

WHEN FLAKY-TEST PREDICTORS FAIL TO TRANSFER: RANKING COLLAPSE UNDER CROSS-PROJECT DEPLOYMENT

Muhammad Arif Shah[*1], **Saba Gul**[2]

[*1,2]*School of Computing, Pak-Austria Fachhochschule Institute of Applied Sciences and Technology*

[*1]*arif.websol@gmail.com*

DOI: https://doi.org/10.5281/zenodo.23389916

Keywords
Flaky tests; flaky-test prediction; cross-project prediction; leave-one-project-out evaluation; ranking quality; decision thresholds; independent replication; empirical software engineering.

Article History
Received: 01 January 2026
Accepted: 16 March 2026
Published: 31 March 2026

Copyright @Author
Corresponding Author: *
Muhammad Arif Shah

Abstract

Flaky-test predictors built on static test-code metrics can score highly when they are trained and evaluated within a single project, yet the situation that motivates cross-project prediction, a project with too little labelled history of its own, requires a model to be deployed on a project it has never seen. This study audits that deployment step on the public 29-project cross-project flaky-test dataset (87,875 tests, 1,385 of them flaky), audited at a fixed repository commit, using standardized logistic regression (LR), random forest (RF), and histogram gradient boosting (HGB) classifiers with 25 static predictors. A five-project discovery stage tested a pre-specified feature-family fragility hypothesis. That hypothesis failed (1 of 5 projects passed; 3 were required), but the discovery data showed a large collapse from within-project to leave-one-project-out (LOPO) performance. The study was therefore redesigned, and the redesigned hypotheses were tested on the 24 remaining projects, with all discovery projects excluded from both training and targets and with decision rules frozen before any replication outcome was viewed. Cross-project transfer failure replicated: the median within-project minus LOPO flaky-class F1 was 0.800 for RF and 0.720 for HGB, meeting the frozen criterion, whereas the LR drop of 0.093 was statistically significant but fell below the frozen 0.10 effect threshold. The loss was dominated by ranking collapse. Median PR-AUC fell from 0.343 to 0.089 (LR), from 0.935 to 0.025 (RF), and from 0.903 to 0.035 (HGB), with losses in 23 or 24 of 24 projects (Holm-adjusted $p \leq 2.5 \times 10^{-6}$). Decision thresholds selected exclusively on source projects removed the zero-positive output that RF produced at threshold 0.50 in 23 of 24 targets, but median F1 gains were only −0.004 (LR), 0.020 (RF), and 0.001 (HGB), and none met the frozen 0.05 criterion. A post-confirmatory robustness stage found the ranking collapse for all three models under repeated within-project baselines, among targets with at least 30 flaky tests, in both low- and high-prevalence strata, and under Recall@K and Recall@1% review budgets. Alternative source-only threshold objectives did not restore performance, and static feature-mean shift was not associated with the size of the loss. For these predictors and this benchmark, strong within-project scores did not indicate usable cross-project rankings, and threshold recalibration alone could not repair a ranking that had not transferred.

1. INTRODUCTION

A flaky test is a test that can pass and fail on the same version of the code. Flaky tests waste developer time, delay continuous integration, and erode trust in test outcomes, because a failure that may be spurious invites developers to ignore failures in general (Luo et al. 2014; Eck et al. 2019; Parry et al. 2022). The direct way to identify flaky tests is to rerun tests many times and observe inconsistent outcomes, but rerunning is expensive and may still miss rarely manifesting flakiness. This cost has motivated approaches that detect flakiness without exhaustive reruns, either by using information from a single execution (Bell et al. 2018) or by predicting flakiness from properties of the test and the code it exercises (Pinto et al. 2020; Alshammari et al. 2021; Verdecchia et al. 2021; Fatima et al. 2023). Static predictors, which use only metrics computed from source code, are particularly attractive because they require no test execution at all (Pontillo et al. 2021, 2022).

Most flaky-test prediction studies evaluate a model on data drawn from the same projects that supplied its training data. A project that would benefit most from a predictor, however, is often one that has not yet accumulated a labelled history of flaky tests. For such a project the predictor must be trained on other projects and applied to the new one. Afeltra et al. (2024) studied this cross-project setting on a dataset of 1,385 flaky tests from 29 open-source projects and found that cross-project prediction was difficult, that filtering approaches helped, and that the TrAdaBoost transfer-learning method reduced data heterogeneity. Their study established cross-project flaky-test prediction as a research problem in its own right and made the dataset used here publicly available.

A cross-project performance score, however, does not reveal what has gone wrong when it is low. Two failure modes have different consequences and different remedies. In the first, the model still orders target tests sensibly, assigning higher scores to flaky tests than to most non-flaky tests, but its scores are shifted relative to the target project, so that a fixed decision threshold such as 0.50 flags too many or too few tests. This is a threshold failure, and it can in principle be repaired by choosing a better threshold. In the second, the model's scores no longer separate flaky from non-flaky tests in the target project at all. This is a ranking failure, and no threshold can repair it, because every threshold cuts the same uninformative ordering. Distinguishing the two matters operationally. A team that receives a ranked list of tests to inspect, or that quarantines the top-scoring tests, depends on the ranking; a team that acts on binary flags depends on both the ranking and the threshold. The distinction also determines whether simple deployment-time fixes, such as re-tuning the threshold on available source data, are worth attempting.

This paper reports a staged reliability audit of static flaky-test prediction under cross-project deployment. It compares within-project evaluation, in which a model is trained and tested on disjoint tests of the same project, with strict leave-one-project-out (LOPO) evaluation, in which the target project contributes nothing to training. It separates ranking loss, measured by PR-AUC and by review-budget metrics, from threshold loss, measured by the change in flaky-class F1 when the decision threshold is selected exclusively from source-project out-of-fold predictions. No target label is used for any training or threshold decision. The project, not the individual test, is the unit of inference throughout.

The audit did not begin with this question, and the paper reports how it arrived there. The original study, frozen before any model was trained, tested whether cross-project flaky-test prediction depends fragilely on particular families of static features. That hypothesis failed in a five-project discovery stage: only one of five discovery projects met the frozen fragility rule, against the three that were required. The same discovery data, however, showed that models that performed strongly within projects collapsed under LOPO deployment. The study was redesigned around this observation, and the redesigned hypotheses were then tested on the 24 projects not used for discovery, with the five discovery projects excluded from both training and targets and with all decision rules frozen before any replication

outcome was viewed. A further post-confirmatory stage examined whether the replicated effect was robust to the choice of within-project baseline, to the number of flaky tests in the target, to target prevalence, to the choice of ranking metric, and to the threshold-selection objective, and it examined calibration and static feature shift as diagnostic context.

The independent replication confirmed a large transfer failure for random forest and histogram gradient boosting and a smaller, statistically significant degradation for logistic regression that did not reach the frozen effect-size criterion. For all three classifiers the degradation was dominated by ranking collapse: median PR-AUC fell by between 0.26 and 0.84, in at least 23 of 24 projects. Source-only threshold selection removed a conspicuous failure mode, the complete absence of positive predictions by random forest at threshold 0.50 in 23 of 24 targets, but recovered very little target F1, because the ranking it thresholded was already uninformative. These findings held in every robustness analysis.

The contributions of the paper are as follows. (i) It provides a staged reliability audit of static flaky-test prediction under cross-project deployment that separates discovery, independent replication on 24 held-out projects, and post-confirmatory robustness analysis, each under a frozen protocol. (ii) It decomposes cross-project degradation into a ranking component and a threshold component and shows that, for the classifiers studied, ranking collapse dominates. (iii) It evaluates threshold adaptation that uses source projects only, without target-label leakage, and shows that it removes zero-positive collapse but yields little recovery of target F1, a result that holds for three threshold objectives. (iv) It reports operational ranking metrics (Recall@K and Recall@1%), positive-count and prevalence sensitivity, calibration, and static feature-shift diagnostics for the cross-project setting. (v) It documents a transparent negative result: the initial feature-fragility hypothesis failed, and the paper reports that failure and the discovery-driven redesign that followed it. The paper does not propose a new predictor and does not claim that cross-project flaky-test prediction is infeasible in general.

Section 2 reviews related work, and Section 3 states the research questions. Section 4 describes the study design, including the outcome of the discovery stage. Section 5 reports the results by research question, and Section 6 discusses them. Section 7 summarises practical implications, Section 8 discusses threats to validity, Section 9 describes data and code availability, and Section 10 concludes.

2 Background and Related Work

2.1 Flaky tests and their detection

Luo et al. (2014) analysed fixes of flaky tests in open-source projects and identified recurring root causes, among them asynchronous waiting, concurrency, and dependence on test execution order. Eck et al. (2019) complemented this view by studying how developers perceive and fix flaky tests, and Parry et al. (2022) surveyed the literature on the causes, costs, detection, and mitigation of flakiness. Detecting flaky tests by rerunning is reliable only when enough reruns are performed, which makes it costly in large test suites. DeFlaker (Bell et al. 2018) reduces this cost by marking a failing test as flaky when it fails without executing any changed code. Other work predicts flakiness from features of the test, sometimes combined with dynamic information: FlakeFlagger (Alshammari et al. 2021) combines static and execution-derived features, and Flakify (Fatima et al. 2023) applies a pre-trained language model to the source code of test cases.

2.2 Static flaky-test prediction

Pinto et al. (2020) showed that the vocabulary of test code can distinguish flaky from non-flaky tests with high accuracy on their benchmark. Haben et al. (2021) replicated this approach under a stricter, time-aware evaluation and on a new Python dataset, and reported that the stricter validation lowered performance while the main conclusions still held. Verdecchia et al. (2021) proposed a fast similarity-based static predictor, and Camara et al. (2021) examined test smells as predictors of flakiness. Pontillo et al. (2021) studied the

feasibility of predicting flakiness from static test and production code metrics and test smells, and extended this line in a larger study of how far static prediction can go (Pontillo et al. 2022); the topic was also summarised in Pontillo (2022). The predictor set used in the present study, which combines test size and complexity, test smells, object-oriented structure, Halstead measures, and design smells, belongs to this static family. These studies collectively show that static features carry signal about flakiness; they do not, by themselves, show whether that signal is portable across projects.

2.3 Cross-project flaky-test prediction

Afeltra et al. (2024) conducted a large-scale empirical investigation of cross-project flaky-test prediction on 1,385 flaky tests from 29 projects. They evaluated static prediction models in a cross-project setting together with feature-based and instance-based filtering methods, and reported that cross-project prediction is challenging, that filtering improves performance, and that the TrAdaBoost method, which reduced data heterogeneity, reached an F-measure of about 70%. TrAdaBoost, as originally formulated, reweights a labelled sample from the target domain alongside source data (Dai et al. 2007). The present study uses the dataset released with that work but addresses a different question. It does not propose or compare transfer-learning or filtering methods, and it does not use target labels in any form. Instead, it asks how much of the performance observed within projects survives a strict deployment in which the target project is entirely unseen, and whether any loss is attributable to ranking or to thresholds. The two studies therefore answer complementary questions, and the results below should not be read as a re-evaluation of the methods studied by Afeltra et al. (2024).

2.4 Reliability under project and distribution shift

Cross-project prediction has a long history in defect prediction. Zimmermann et al. (2009) found that cross-project defect prediction succeeded in only a small fraction of project pairs, and Turhan et al. (2009) compared the value of cross-company and within-company data. Later benchmarks and meta-analyses (Herbold et al. 2018; Hosseini et al. 2019) showed that cross-project performance depends strongly on the approach, the data, and the evaluation design. In machine learning, such failures are studied under the heading of dataset shift, which distinguishes changes in the input distribution, in class priors, and in the relationship between inputs and labels (Moreno-Torres et al. 2012). A change in class prior alone leaves the ranking produced by a well-specified model largely intact while shifting its calibrated probabilities, which is why prior-adjustment and threshold-adjustment procedures exist (Saerens et al. 2002). A change in the input–label relationship can instead destroy the ranking. Distinguishing ranking quality from threshold-dependent performance therefore helps to identify which kind of shift a predictor has suffered.

Evaluation choices also matter under strong class imbalance. Accuracy is uninformative when the positive class is rare, PR-AUC is more informative than ROC-AUC about performance on the rare class (Saito and Rehmsmeier 2015), and the Matthews correlation coefficient summarises a confusion matrix more faithfully than F1 or accuracy (Chicco and Jurman 2020). Model-validation choices change performance estimates in software analytics (Tantithamthavorn et al. 2017), and leakage between training and evaluation data can inflate them (Kapoor and Narayanan 2023). Calibration, the agreement between predicted probabilities and observed frequencies, is usually assessed with the Brier score (Brier 1950) and with binned calibration error (Naeini et al. 2015; Guo et al. 2017). The present study draws on these strands to separate ranking, threshold, and calibration effects in the cross-project flaky-test setting.

3 Research Questions

The study addresses five research questions. RQ1 to RQ3 were answered by the independent replication stage under frozen decision rules. RQ4

and RQ5 were answered by the post-confirmatory robustness stage.

RQ1. How much does flaky-test prediction performance change when models move from within-project evaluation to true cross-project LOPO deployment? This question tests whether the transfer collapse observed during discovery replicates on projects that were not used to formulate it.

RQ2. Does cross-project degradation reflect loss of ranking quality or primarily a mismatch in the default decision threshold? This question compares the threshold-free PR-AUC loss with the threshold-dependent F1 loss.

RQ3. Can decision thresholds learned exclusively from source projects recover target-project flaky-test detection performance? This question tests the simplest deployment-time remedy that does not require target labels.

RQ4. Does cross-project ranking degradation remain under repeated within-project baselines, positive-count sensitivity, and target-prevalence stratification? This question tests whether the RQ2 result depends on a single within-project split, on targets with very few flaky tests, or on target prevalence.

RQ5. How does cross-project transfer affect operational ranking quality, calibration, and its relationship with source–target static-feature shift? This question translates ranking loss into review-budget terms and examines diagnostic correlates of the loss.

4 Study Design

The study was organised in four stages: a Stage 0 audit of the dataset, a Stage 1 discovery stage, a Stage 2 independent replication, and a Stage 3 robustness and mechanism analysis. Each modelling stage was governed by a protocol that was frozen before the stage was executed. Table 4 summarises the design.

4.1 Dataset and Stage 0 audit

The study uses the dataset distributed in the replication repository of Afeltra et al. (2024) (https://github.com/alfcan/crossproject-flaky-test-prediction). The repository was cloned and checked out at commit 8fe244f03122706763484dea7b4d98b09fefb923, and the SHA-256 digest of the file dataset.csv at that commit was 6c2066eb1bf1824887a72c6f6df3cb5106894efde206686eb564287fe6547e50. Every later stage re-cloned the repository, verified the commit and the digest, and reproduced the Stage 0 counts before any model was trained.

Stage 0 performed a schema and semantics audit without training any classifier. The file contains 87,875 test cases from 29 projects in 28 columns: a project name (nameProject), a test-case identifier (testCase), 25 numeric static predictors, and a binary label (isFlaky). The audit found 1,385 flaky and 86,490 non-flaky tests, an overall flaky prevalence of 1.58%, no missing values, and no exact duplicate rows, and it resolved nameProject as the grouping variable and isFlaky as the target. The testCase field is nearly unique (86,200 distinct values) and was flagged as an identity field. Neither nameProject nor testCase was ever used as a predictor. The audit concluded with the status STAGE0_PASS_SCHEMA_AND_SEMANTICS. Table 1 lists the projects. Project names are shown without the commit suffix that the dataset appends to them.

Table 1 Projects in the audited dataset. "Prev." is the flaky prevalence. The prevalence stratum (median split at 1.94%) and the positive-count subset (≥ 30 flaky tests) are defined over the 24 replication projects and are used in Stage 3.

Project	Tests	Flaky	Prev. (%)	Role	Prevalence stratum	≥ 30 flaky
Dubbo	1,639	139	8.48	Discovery	–	–
Hadoop	13,662	177	1.30	Discovery	–	–
Ignite-3	2,173	118	5.43	Discovery	–	–

Project	Tests	Flaky	Prev. (%)	Role	Prevalence stratum	≥ 30 flaky
NiFi	11,145	138	1.24	Discovery	–	–
ORMLite Core	1,042	90	8.64	Discovery	–	–
Achilles	401	24	5.99	Replication	High	No
Activiti	964	22	2.28	Replication	High	No
admiral	2,390	55	2.30	Replication	High	Yes
adyen-java-api-library	253	45	17.79	Replication	High	Yes
aem-core-wcm-components	1,092	24	2.20	Replication	High	No
Chronicle-Wire	460	35	7.61	Replication	High	Yes
commons-lang	2,711	25	0.92	Replication	Low	No
DataflowTemplates	694	39	5.62	Replication	High	Yes
druid	10,729	30	0.28	Replication	Low	Yes
fastjson	1,860	25	1.34	Replication	Low	No
flink	9,254	23	0.25	Replication	Low	No
graylog2-server	2,477	21	0.85	Replication	Low	No
http-request	184	28	15.22	Replication	High	No
jackrabbit-oak	10,101	20	0.20	Replication	Low	No
Java-WebSocket	419	28	6.68	Replication	High	No
mockserver	2,104	27	1.28	Replication	Low	No
nacos	1,850	26	1.41	Replication	Low	No
ozone	2,480	24	0.97	Replication	Low	No
servicecomb-java-chassis	2,423	41	1.69	Replication	Low	Yes
typescript-generator	132	34	25.76	Replication	High	Yes
visualee	134	43	32.09	Replication	High	Yes
vpc-java-sdk	2,551	22	0.86	Replication	Low	No
wildfly	910	37	4.07	Replication	High	Yes
wro4j	1,641	25	1.52	Replication	Low	No
Total (29 projects)	**87,875**	**1,385**	**1.58**			

4.2 Predictors and classifiers

All models used exactly the 25 audited static predictors. For the discovery stage these were grouped, before any model was trained, into five families (Table 2): test size and complexity, test smells and assertions, object-oriented structure, Halstead measures, and design smells.

Table 2 Static predictors and the feature families defined in the frozen Stage 1 protocol.

Family	Predictors (n)	Predictors
TEST_SIZE_ COMPLEXITY	2	tloc, tmcCabe
TEST_SMELLS_ ASSERTIONS	8	assertionDensity, assertionRoulette, mysteryGuest, eagerTest, sensitiveEquality, resourceOptimism, conditionalTestLogic, fireAndForget
STRUCTURAL_OO	7	loc, lcom2, lcom5, cbo, wmc, rfc, mpc
HALSTEAD	3	halsteadVocabulary, halsteadLength, halsteadVolume
DESIGN_SMELLS	5	classDataShouldBePrivate, complexClass, spaghettiCode, functionalDecomposition, godClass
All predictors	25	

Three classifiers were used in every stage, with fixed configurations and no hyperparameter tuning, implemented in scikit-learn (Pedregosa et al. 2011). Logistic regression (LR) standardised the predictors and used the liblinear solver with balanced class weights. Random forest (RF; Breiman 2001) used 200 trees, square-root feature sampling, a minimum of two samples per leaf, and balanced-subsample class weights. Histogram gradient boosting (HGB), a histogram-based implementation of gradient boosting (Friedman 2001), used a learning rate of 0.05, 200 iterations, at most 15 leaf nodes, L2 regularisation of 1.0, and balanced sample weights. A fixed random state of 42 was used for model fitting. The configurations are listed in Appendix B.

4.3 Original discovery stage and its outcome

Protocol. Stage 1 was designed to test a feature-fragility hypothesis: that cross-project flaky-test prediction depends on a small, unstable subset of static feature families. The five projects with the largest numbers of flaky tests were chosen as discovery projects before any model outcome was available: Hadoop, Dubbo, NiFi, Ignite-3, and ORMLite Core (662 flaky tests in total). Each model was evaluated within project, using a grouped and stratified 80/20 split, and under LOPO across the five discovery projects, which was the primary setting. Under LOPO, each discovery target was predicted by a model trained on the other four. Each family was removed in turn, and the model was retrained. A model showed a fragility signal for a project if removing the same family caused, simultaneously, an absolute flaky-class F1 drop of at least 0.05, a relative PR-AUC drop of at least 15%, and prediction churn of at least 0.05. A project passed if at least two of the three models showed a signal, and the stage would proceed (GO) if at least three of the five projects passed.

Outcome. The stage returned STOP_OR_REDESIGN. Only Ignite-3 passed, with fragility signals for LR and HGB. ORMLite Core showed a signal for LR alone, and Hadoop, Dubbo, and NiFi showed none (Table 3). TEST_SIZE_COMPLEXITY was the family with the largest F1 drop in 11 of the 15 project–model pairs, but this concentration did not translate into the joint fragility pattern that the frozen rule required. The original hypothesis is therefore reported as not supported.

Table 3 Stage 1 discovery outcome under the frozen feature-fragility rule (LOPO over the five discovery projects). A model shows a signal if removing the same feature family reduced flaky-class F1 by ≥ 0.05, reduced PR-AUC by ≥ 15% (relative), and changed ≥ 5% of predictions.

Discovery project	LR signal	RF signal	HGB signal	Models with signal	Project decision
Hadoop	No	No	No	0/3	Fail
Dubbo	No	No	No	0/3	Fail
NiFi	No	No	No	0/3	Fail
Ignite-3	Yes	No	Yes	2/3	Pass
ORMLite Core	Yes	No	No	1/3	Fail

Stage 1 verdict: STOP_OR_REDESIGN (1/5 projects passed; at least 3/5 required).

What the discovery data showed instead. The ablation analysis was run on LOPO predictions, and the LOPO baseline was itself very weak. Across the five discovery projects, the median within-project flaky-class F1 was 0.347 for LR, 0.767 for RF, and 0.714 for HGB, whereas the median LOPO F1 at threshold 0.50 was 0.105, 0.000, and 0.006. Median PR-AUC fell from 0.282 to 0.072 for LR, from 0.788 to 0.085 for RF, and from 0.765 to 0.085 for HGB (Fig. 1). A model whose LOPO F1 is already close to zero leaves little room for a further 0.05 drop when a feature family is removed, so the failure of the fragility rule may partly reflect this floor. The more consequential observation was the collapse itself: classifiers that separated flaky from non-flaky tests well within projects lost most of that ability when deployed on an unseen project.

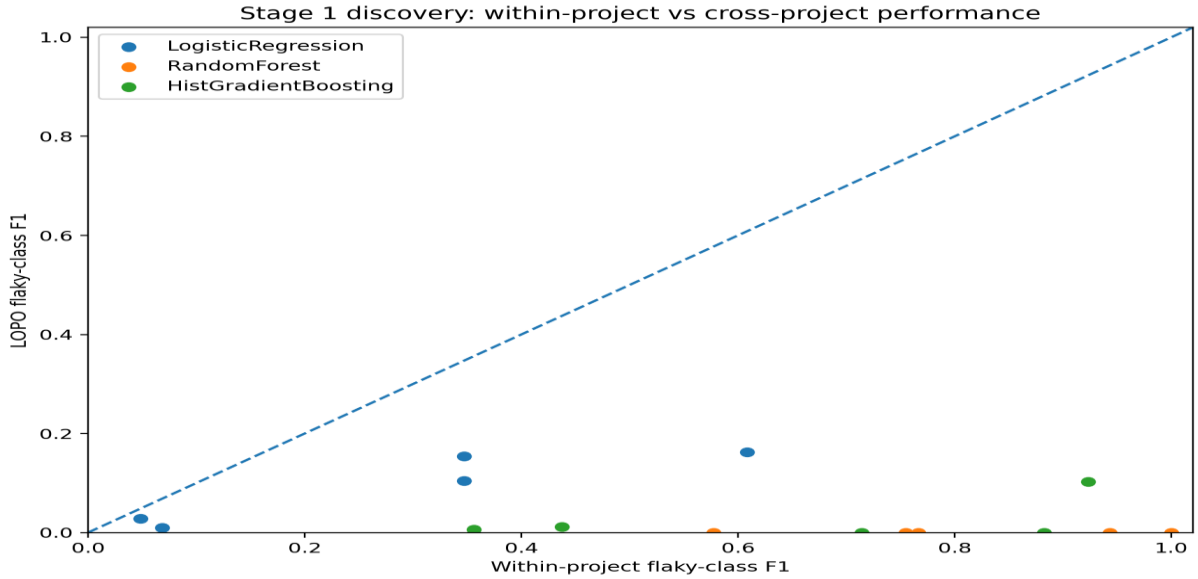


Fig. 1 Stage 1 discovery (5 projects, exploratory): within-project flaky-class F1 (x-axis) against LOPO flaky-class F1 at threshold 0.50 (y-axis) for each project and classifier. Points below the dashed diagonal indicate lower performance under LOPO. This observation motivated the redesigned Stage 2 and is not itself confirmatory evidence.

4.4 Redesigned independent replication (Stage 2)

Rationale and independence. The redesign was driven by the discovery data, so testing it on the same projects would have been circular. The five discovery projects were therefore used only to generate the hypothesis and were excluded completely from Stage 2: they were neither targets nor training data. The remaining 24 projects (58,214 tests, 723 flaky) formed the replication

pool. For each replication target, the LOPO model was trained on the other 23 replication projects only. All Stage 2 decision rules were frozen before any of the 24 replication-project outcomes was inspected. Stage 2 is the only stage that provides independent confirmatory evidence in this paper.

Within-project evaluation. For each replication project, the tests were split by a stratified, grouped five-fold partition (StratifiedGroupKFold, shuffled, random state 42) with testCase as the grouping key, and the first fold was used as an approximately 20% test set. The model was trained on the remaining tests of the same project and evaluated at threshold 0.50. Within-project test sets were small, containing between 3 and 11 flaky tests (median 5).

LOPO evaluation. For each target, the model was trained on all tests of the 23 other replication projects (for example, 57,813 tests including 699 flaky tests when Achilles was the target) and applied to all tests of the target. Predictions were evaluated at the default threshold of 0.50 and at a source-only threshold (Section 4.5). The ranking metrics do not depend on the threshold.

Frozen decision rules. Three confirmatory comparisons were made for each model, giving nine tests in total. *Transfer failure* was declared replicated if at least two of the three models satisfied all of the following: median within-project F1 minus LOPO default-threshold F1 ≥ 0.10; a positive difference in more than half of the 24 projects; and Holm-adjusted $p < 0.05$. A *threshold component* was declared present if at least two models showed a median source-only-threshold F1 minus default-threshold F1 ≥ 0.05 with Holm-adjusted $p < 0.05$. A *ranking component* was declared present if at least two models showed a median within-project PR-AUC minus LOPO PR-AUC ≥ 0.05 with Holm-adjusted $p < 0.05$. The two mechanism rules were evaluated independently, so the mechanism verdict could be threshold dominant, ranking dominant, mixed, transfer failure without either frozen mechanism, or no replicated transfer failure.

4.5 Source-only threshold adaptation

The source-only threshold was selected without access to the target. For each target and model, the 23 source projects were partitioned into five project-grouped folds (GroupKFold), and out-of-fold probabilities were generated for every source test by models trained on the other source folds. Candidate thresholds were 0.50 together with the unique values of 500 evenly spaced quantiles (0.001 to 0.999) of the out-of-fold probabilities. The threshold that maximised flaky-class F1 on the source out-of-fold predictions was selected, with ties broken toward the higher threshold. The model was then refitted on all 23 source projects and the frozen threshold applied to the untouched target. Because the out-of-fold folds are grouped by project, the threshold reflects how the model behaves on projects it has not seen, which is the deployment condition. Target labels were never used.

This procedure also allows the study to record *zero-positive collapse*, defined as a target for which a model flags no test at all. A zero-positive output yields a flaky-class F1 of zero regardless of the ranking, so it is an important practical failure in its own right.

4.6 Robustness and mechanism analyses (Stage 3)

Stage 3 was a post-confirmatory robustness and mechanism analysis governed by its own frozen protocol. It used the same 24 replication projects, and the five discovery projects remained excluded. Because it reuses the Stage 2 projects, Stage 3 is not a second independent replication. It comprised the following analyses.

(1) *Repeated within-project baselines.* The grouped 80/20 within-project evaluation was repeated with three seeds (17, 42, and 73), and the per-project median across seeds was used as the within-project value. Test sets contained between 3 and 12 flaky tests.

(2) *Operational ranking metrics.* In addition to PR-AUC, Recall@K, Precision@K, and Recall@1% were computed. K equals the number of truly flaky tests in the evaluated set, so Recall@K and Precision@K coincide; Recall@1% is the fraction

of flaky tests found among the top-ranked 1% of tests (at least one test).

(3) *Positive-count sensitivity.* The PR-AUC loss was recomputed for the nine targets with at least 30 flaky tests.

(4) *Prevalence strata.* The 24 targets were split at the median prevalence (1.94%) into low- and high-prevalence strata of 12 projects each.

(5) *Alternative threshold objectives.* Source-only thresholds were selected, as in Section 4.5, to maximise F1, the Matthews correlation coefficient (MCC), or balanced accuracy.

(6) *Calibration.* The Brier score and the expected calibration error over ten equal-width probability bins (ECE-10) were computed within project and under LOPO.

(7) *Static feature shift.* For each target, the absolute standardised mean difference (SMD) between source and target was computed for each of the 25 predictors, using the pooled standard deviation (Austin 2009), and summarised by its median across predictors.

Frozen robustness criterion. Ranking collapse was declared robust if at least two of the three models satisfied all three of the following: (A) median repeated-within-project PR-AUC minus LOPO PR-AUC ≥ 0.05 with Holm-adjusted $p < 0.05$; (B) median PR-AUC loss ≥ 0.05 among targets with at least 30 flaky tests; and (C) median PR-AUC loss > 0 in both the low- and the high-prevalence stratum. The alternative-threshold, calibration, and shift analyses were declared diagnostic only and could not change the Stage 2 verdict.

Table 4 Summary of the staged design. Every modelling protocol was frozen before the corresponding stage was executed.

Stage	Projects	Evaluation	Frozen decision rule	Outcome
0 Audit	All 29	Schema, counts, missingness, duplicates, identifiers	Commit and SHA-256 verified; documented counts reproduced	STAGE0_PASS_ SCHEMA_AND_ SEMANTICS
1 Discovery	Hadoop, Dubbo, NiFi, Ignite-3, ORMLite Core	Within-project 80/20; LOPO over the 5 projects; feature-family ablation	Same family causes ΔF1 ≥ 0.05, relative ΔPR-AUC ≥ 15%, churn ≥ 0.05; ≥ 2/3 models per project; GO if ≥ 3/5 projects	1/5 passed; STOP_ OR_REDESIGN
2 Independent replication	Remaining 24; discovery projects excluded from training and targets	Within-project 80/20; LOPO on 23 sources; default and source-only thresholds	Transfer, threshold, and ranking rules (Section 4.4); 9 Wilcoxon tests, Holm-adjusted	REPLICATED_ TRANSFER_ FAILURE; RANKING_ DOMINANT_ FAILURE
3 Robustness	Same 24; discovery projects excluded	Repeated within (3 seeds); LOPO ranking, threshold, calibration, shift diagnostics	Criteria A–C for ≥ 2/3 models (Section 4.6); 9 Wilcoxon tests, Holm-adjusted	ROBUST_ RANKING_ COLLAPSE (3/3 models)

4.7 Metrics and statistical analysis

The primary metrics were flaky-class F1, PR-AUC (computed as average precision), MCC, and balanced accuracy. Precision, recall, ROC-AUC, and macro F1 were recorded as secondary metrics. Accuracy is not reported as a headline metric, because with a flaky prevalence of 1.58% a classifier that never predicts a flaky test would be about 98% accurate. PR-AUC is threshold-free and is the primary measure of ranking quality. Its no-skill baseline equals the prevalence of the evaluated set. Because each within-project test set is a stratified sample of the same project that is used as the LOPO target, the within-project and LOPO PR-AUC values compared for a given project have approximately the same baseline.

The project is the unit of inference. For each model and comparison, the 24 paired project-level differences were tested with the two-sided Wilcoxon signed-rank test (Wilcoxon 1945). In Stage 2, the nine confirmatory p-values (three comparisons × three models) were adjusted together with the Holm procedure (Holm 1979); in Stage 3, the nine ranking tests (three metrics × three models) were adjusted as a separate family. Medians of paired differences and the number of projects with a positive difference are reported as effect measures. With 24 projects, the smallest attainable two-sided p-value is about 1.2×10^{-7}, so very small p-values reflect the consistency of the direction of the effect across projects rather than precision of its size. Confidence intervals were not computed. Spearman correlations in Stage 3 are reported descriptively, without multiplicity adjustment.

All analyses were executed in Python 3.13 with scikit-learn, SciPy, pandas, and NumPy. Each stage wrote metric tables, prediction files, figures, an integrity manifest, and its verdict. All numbers reported in this paper were taken from those outputs.

5 Results

Sections 5.1 to 5.3 report the Stage 2 independent replication (n = 24 projects). Section 5.3 also reports the Stage 3 alternative-threshold diagnostic, and Sections 5.4 and 5.5 report the Stage 3 robustness and mechanism analyses on the same 24 projects. Table 5 gives the median project-level metrics in each Stage 2 evaluation setting, and Table 6 gives the nine confirmatory tests.

IEER
Institute for Excellence in Education & Research

Table 5 Stage 2 independent replication (n = 24 projects): median project-level metrics by classifier and evaluation setting. PR-AUC and ROC-AUC do not depend on the threshold, so they are identical in the two LOPO rows.

Model	Setting	Flaky F1	PR-AUC	MCC	Bal. acc.	Recall	ROC-AUC
LR	Within-project	0.164	0.343	0.226	0.853	0.866	0.915
LR	LOPO, threshold 0.50	0.051	0.089	0.059	0.583	0.376	0.597
LR	LOPO, source-only threshold	0.026	0.089	0.006	0.507	0.038	0.597
RF	Within-project	0.800	0.935	0.806	0.908	0.845	0.996
RF	LOPO, threshold 0.50	0.000	0.025	0.000	0.500	0.000	0.488
RF	LOPO, source-only threshold	0.020	0.025	−0.004	0.495	0.131	0.488
HGB	Within-project	0.764	0.903	0.760	0.911	0.866	0.996
HGB	LOPO, threshold 0.50	0.018	0.035	−0.002	0.499	0.045	0.582
HGB	LOPO, source-only threshold	0.037	0.035	0.008	0.523	0.195	0.582

Table 6 Stage 2 confirmatory tests (n = 24 projects; two-sided paired Wilcoxon signed-rank tests; Holm adjustment across all nine tests). "Median diff." is the median of the project-level paired differences; "Criterion met" applies the frozen effect-size, direction, and Holm conditions of the corresponding rule (Section 4.4).

Comparison	Model	Median diff.	Positive projects	W	p	Holm p	Criterion met
Transfer: within F1 − LOPO F1	LR	0.092916	23/24	14	1.3×10^{-5}	7.9×10^{-5}	No
Transfer: within F1 − LOPO F1	RF	0.800000	23/24	0	2.7×10^{-5}	1.1×10^{-4}	Yes
Transfer: within F1 − LOPO F1	HGB	0.720470	24/24	0	1.8×10^{-5}	9.1×10^{-5}	Yes
Ranking: within PR-AUC − LOPO PR-AUC	LR	0.263841	23/24	2	3.6×10^{-7}	2.5×10^{-6}	Yes
Ranking: within PR-AUC − LOPO PR-AUC	RF	0.838955	24/24	0	1.2×10^{-7}	1.1×10^{-6}	Yes
Ranking: within PR-AUC − LOPO PR-AUC	HGB	0.787104	24/24	0	1.2×10^{-7}	1.1×10^{-6}	Yes
Threshold: tuned F1 − default F1	LR	−0.003556	10/24	84	0.168	0.318	No
Threshold: tuned F1 − default F1	RF	0.019816	18/24	0	2.0×10^{-4}	5.9×10^{-4}	No
Threshold: tuned F1 − default F1	HGB	0.000957	13/24	75	0.159	0.318	No

IEER
Institute for Excellence in Education & Research

5.1 RQ1: Within-project versus cross-project performance

All three classifiers performed markedly worse under LOPO deployment than within project. RF and HGB attained high within-project scores, with median flaky-class F1 of 0.800 and 0.764, median MCC of 0.806 and 0.760, and median PR-AUC of 0.935 and 0.903. Under LOPO at threshold 0.50 their median F1 fell to 0.000 and 0.018, their median MCC to 0.000 and −0.002, and their median balanced accuracy to 0.500 and 0.499, the values expected from a classifier with no discriminative ability (Table 5). The median within-project minus LOPO F1 difference was 0.800 for RF (positive in 23 of 24 projects; Holm-adjusted $p = 1.1 \times 10^{-4}$) and 0.720 for HGB (24 of 24; Holm-adjusted $p = 9.1 \times 10^{-5}$). The interquartile ranges of the project-level F1 drops were 0.587–0.878 for RF and 0.504–0.844 for HGB, so large drops were the norm rather than the effect of a few projects. The single RF project without a positive difference, ozone, had an F1 of zero in both settings.

LR behaved differently. Its within-project performance was much lower (median F1 0.164, PR-AUC 0.343), mainly because of low precision at threshold 0.50 (median 0.091) combined with high recall (0.866). Its LOPO F1 at threshold 0.50 fell to a median of 0.051, and the median drop of 0.092916 was positive in 23 of 24 projects with Holm-adjusted $p = 7.9 \times 10^{-5}$. The degradation was therefore statistically significant and consistent in direction, but its median size was below the frozen practical threshold of 0.10, so LR did not satisfy the frozen transfer rule. The

exception to the direction was vpc-java-sdk, where LR's LOPO F1 exceeded its within-project F1.

Two of the three classifiers met all conditions of the frozen transfer rule, so the Stage 2 transfer verdict was REPLICATED_TRANSFER_FAILURE. Figure 2 shows the project-level picture: almost every RF and HGB point lies far below the diagonal, near zero LOPO F1, whereas the LR points are closer to the diagonal because their within-project F1 was already low.

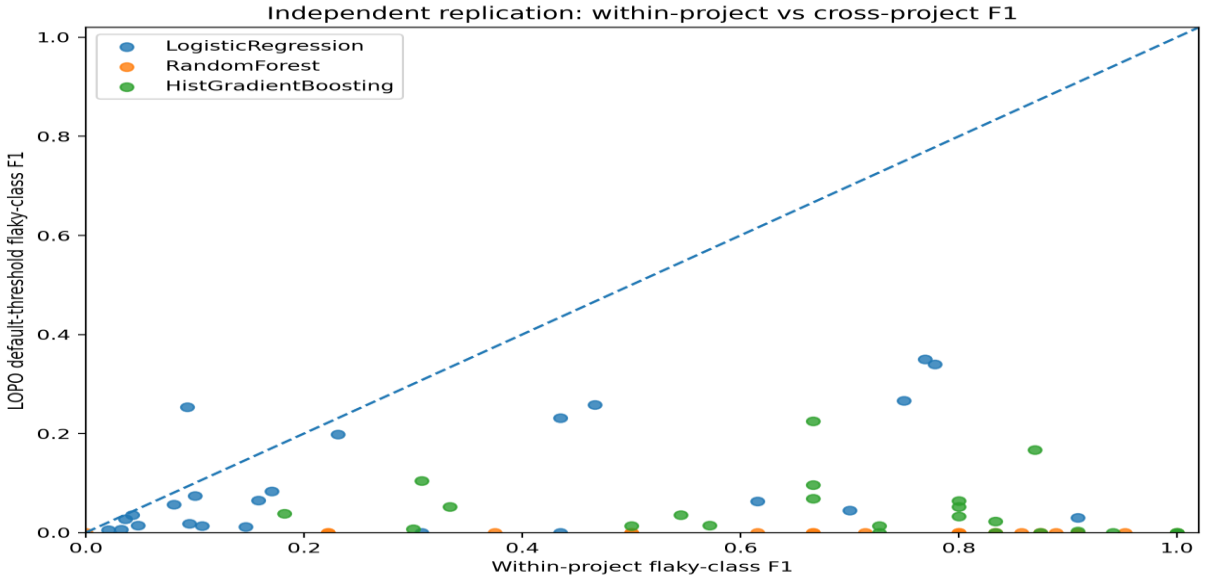


Fig. 2 Stage 2 independent replication (n = 24 projects): within-project flaky-class F1 (x-axis) against LOPO flaky-class F1 at threshold 0.50 (y-axis), one point per project and classifier. Points below the dashed diagonal indicate lower performance under LOPO.

Answer to RQ1. On 24 projects that were not used to formulate the hypothesis, cross-project LOPO deployment reduced flaky-class F1 by a median of 0.800 (RF) and 0.720 (HGB), in at least 23 of 24 projects, satisfying the frozen replication rule. LR degraded significantly and consistently, by a median of 0.093, which was below the frozen 0.10 effect criterion. Transfer failure therefore replicated for two of three classifiers.

5.2 RQ2: Ranking loss versus threshold mismatch

PR-AUC does not depend on a threshold, so a large PR-AUC loss indicates that the model's ordering of target tests has deteriorated. PR-AUC fell for every classifier (Table 5 and Table 6). The median within-project minus LOPO PR-AUC difference was 0.263841 for LR (positive in 23 of 24 projects; Holm-adjusted $p = 2.5 \times 10^{-6}$), 0.838955 for RF (24 of 24; Holm-adjusted $p = 1.1 \times 10^{-6}$), and 0.787104 for HGB (24 of 24; Holm-adjusted $p = 1.1 \times 10^{-6}$). The interquartile ranges of the project-level PR-AUC losses were 0.109–0.456 (LR), 0.590–0.911 (RF), and 0.585–0.863 (HGB). The one LR project with a PR-AUC gain was Chronicle-Wire. All three classifiers met the frozen ranking rule.

The residual LOPO ranking was weak in absolute terms. The median LOPO PR-AUC was 0.089 (LR), 0.025 (RF), and 0.035 (HGB). Relative to the prevalence of each target, which is the PR-AUC of a random ranking, the median ratio of LOPO PR-AUC to target prevalence was 1.68 for LR, 1.34 for RF, and 1.70 for HGB, and LOPO PR-AUC was below the target prevalence in 3, 7, and 2 of the 24 projects. The secondary metric ROC-AUC, which is insensitive to prevalence, tells the same story: the median LOPO ROC-AUC was 0.597 (LR), 0.488 (RF), and 0.582 (HGB), against within-project medians of 0.915, 0.996, and

0.996, and LOPO ROC-AUC was below 0.5 in 4, 13, and 9 of the 24 targets. For RF in particular, the cross-project ranking was, at the median, no better than chance.

For RF and HGB the PR-AUC loss was at least as large as the F1 loss (Fig. 3), which indicates that the drop in hard-decision performance is accounted for by the collapse of the ranking rather than by an otherwise intact ranking cut at a poor threshold. For LR, the PR-AUC loss (0.264) was considerably larger than the F1 loss (0.093). LR's F1 was low in both settings, so F1 understates how much ranking quality LR lost. Combined with the threshold result reported in Section 5.3, in which no model met the frozen threshold rule, the Stage 2 mechanism verdict was RANKING_DOMINANT_FAILURE.

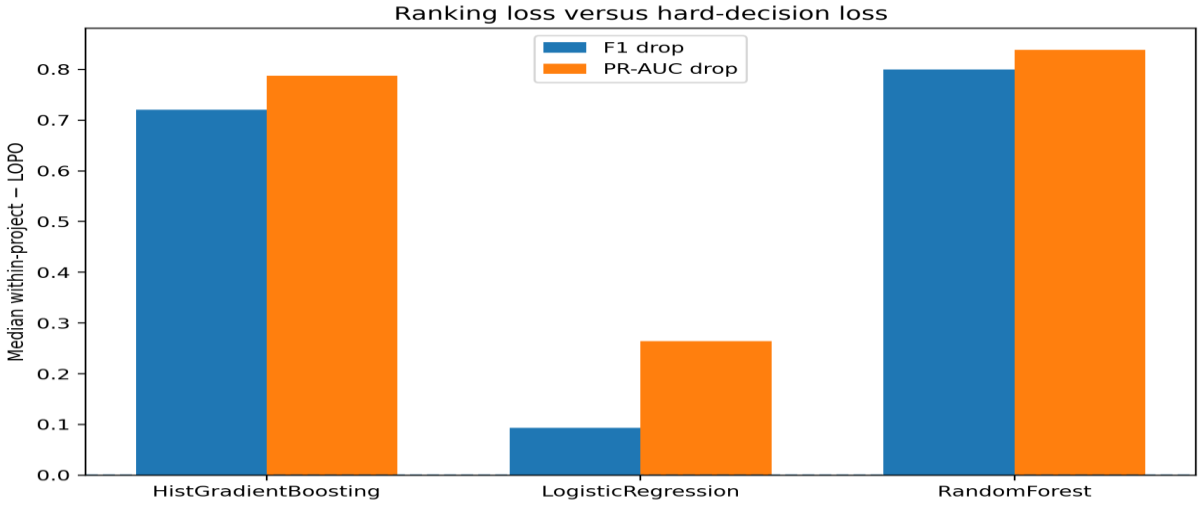


Fig. 3 Stage 2 independent replication: median project-level loss from within-project to LOPO evaluation in flaky-class F1 at threshold 0.50 (hard decisions) and in PR-AUC (ranking), by classifier.

Answer to RQ2. The cross-project degradation is dominated by loss of ranking quality. Median PR-AUC losses were 0.264 (LR), 0.839 (RF), and 0.787 (HGB), in at least 23 of 24 projects, and median LOPO ROC-AUC was close to 0.5. All three classifiers met the frozen ranking rule, and none met the frozen threshold rule, giving a ranking-dominant failure.

5.3 RQ3: Source-only threshold adaptation

The source-only threshold behaved very differently across classifiers (Table 7 and Fig. 5). For RF, the selected thresholds were far below 0.50 (median 0.032; range 0.005–0.079), because RF's out-of-fold probabilities on unseen source projects rarely approached 0.50. At the default threshold, RF flagged no test at all in 23 of the 24 targets, which forces a flaky-class F1 of zero irrespective of the ranking. The source-only threshold removed this zero-positive state in all 23 targets, raising the median number of flagged tests from 0 to 173 and the median recall from 0.000 to 0.131. For HGB, which produced one zero-positive target at 0.50, the median selected threshold was 0.292, and the zero-positive target was also rescued. For LR, the selected thresholds were above 0.50 in every target (median 0.700), because class-weighted LR over-predicted flakiness on unseen source projects; the higher threshold reduced the median number of flagged tests from 341 to 53.5 and the median recall from 0.376 to 0.038.

Table 7 Stage 2 source-only threshold selection and zero-positive collapse (n = 24 targets). "Zero-positive" counts targets in which no test was flagged. The selected threshold is shown as median [minimum, maximum].

"Source OOF best F1" is the F1 achieved by the selected threshold on the project-grouped out-of-fold predictions of the 23 source projects (median over targets).

Model	Zero-pos. at 0.50	Zero-pos. source-only	Rescued	Selected threshold	Source OOF best F1	Recall at 0.50	Recall source-only	Flagged at 0.50	Flagged source-only
LR	0/24	0/24	0	0.700 [0.537, 0.981]	0.045	0.376	0.038	341.0	53.5
RF	23/24	0/24	23	0.032 [0.005, 0.079]	0.033	0.000	0.131	0.0	173.0
HGB	1/24	0/24	1	0.292 [0.029, 0.614]	0.030	0.045	0.195	75.5	215.5

Removing zero-positive collapse did not translate into useful target detection. The median gain in target F1 from the source-only threshold was −0.003556 for LR (positive in 10 of 24 projects; Holm-adjusted p = 0.318), 0.019816 for RF (18 of 24; Holm-adjusted p = 5.9×10^{-4}), and 0.000957 for HGB (13 of 24; Holm-adjusted p = 0.318) (Table 6 and Fig. 4). The RF gain was statistically significant but was well below the frozen practical threshold of 0.05, and the median RF target F1 after adaptation was 0.020. Median MCC after adaptation was 0.006 (LR), −0.004 (RF), and 0.008 (HGB), and median balanced accuracy was 0.507, 0.495, and 0.523. No classifier met the frozen threshold rule.

The source data themselves anticipated this outcome. The best flaky-class F1 attainable by any threshold on the project-grouped out-of-fold predictions of the source projects had a median of only 0.045 for LR, 0.033 for RF, and 0.030 for HGB (Table 7). In other words, when the classifiers were evaluated on source projects that they had not been trained on, their rankings were already nearly uninformative, and threshold selection had little to work with. This diagnostic used no target information.

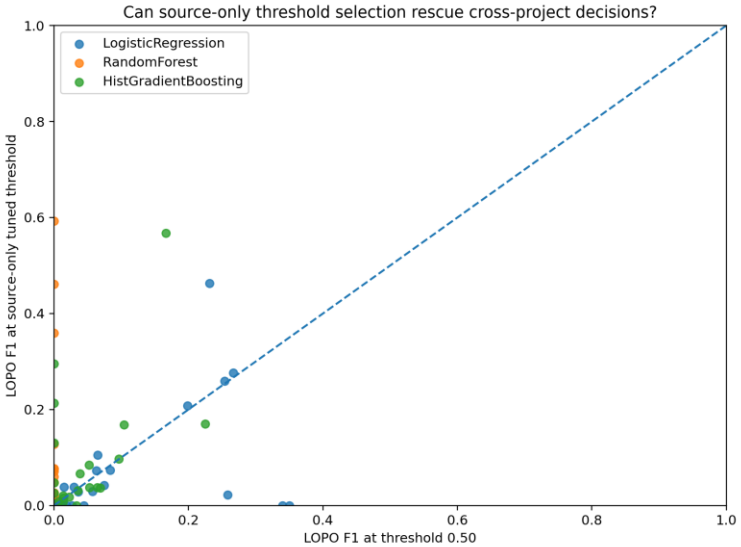


Fig. 4 Stage 2 independent replication: target flaky-class F1 at threshold 0.50 (x-axis) against target F1 at the source-only threshold (y-axis), one point per project and classifier. Points above the diagonal indicate a gain from source-only threshold selection.

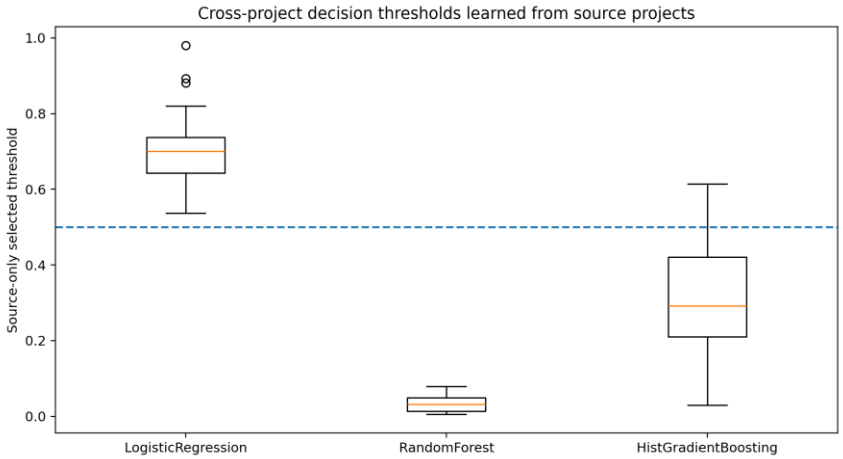


Fig. 5 Distribution of the decision thresholds selected from source-project out-of-fold predictions across the 24 Stage 2 targets. The dashed line marks the default threshold of 0.50.

Alternative threshold objectives (Stage 3 diagnostic). Stage 3 repeated the source-only selection with MCC and balanced accuracy as the objective (Table 8). No objective changed the conclusion. The largest median F1 gains were 0.027681 (MCC) and 0.027838 (balanced accuracy) for RF, and 0.012629 (balanced accuracy) for HGB; for LR all three objectives gave small negative median gains. Median target F1 after adaptation remained below 0.05 for every classifier and objective.

Table 8 Stage 3 diagnostic: median target F1 gain over threshold 0.50 for thresholds selected on source-only out-of-fold predictions with three objectives (n = 24 targets).

Model	Source objective	Median F1 gain	Projects with gain > 0	Median target F1
LR	F1	−0.003556	10/24	0.026
LR	MCC	−0.001495	8/24	0.026
LR	Balanced accuracy	−0.002897	5/24	0.046
RF	F1	0.019816	18/24	0.020
RF	MCC	0.027681	21/24	0.028
RF	Balanced accuracy	0.027838	22/24	0.028
HGB	F1	0.000957	13/24	0.037
HGB	MCC	0.009584	15/24	0.035
HGB	Balanced accuracy	0.012629	17/24	0.040

Answer to RQ3. Thresholds learned only from source projects removed the zero-positive collapse of RF (23 of 23 affected targets) and HGB (1 of 1), but recovered little target F1: median gains were −0.004 (LR), 0.020 (RF), and 0.001 (HGB), and no classifier met the frozen 0.05 criterion. MCC- and balanced-accuracy-based thresholds gave the same picture. Threshold mismatch is operationally visible, especially for RF, but it does not explain the dominant loss, because the ranking that the threshold cuts has already collapsed.

5.4 RQ4: Robustness of the ranking collapse

Repeated within-project baselines. Stage 2 relied on a single within-project split, and within-project test sets contained few flaky tests. Stage 3 therefore repeated the within-project evaluation with three seeds and used the per-project median as the baseline. The median within-project PR-AUC across projects was stable across seeds (LR 0.353, 0.343, and 0.384; RF 0.841, 0.935, and 0.882; HGB 0.865, 0.903, and 0.917 for seeds 17, 42, and 73). Against this repeated baseline, the median PR-AUC loss was 0.256817 for LR, 0.810365 for RF, and 0.781108 for HGB, positive in all 24 projects for every classifier (Holm-adjusted $p = 1.1 \times 10^{-6}$ for each; Table 9 and Fig. 6). The single LR project without a PR-AUC loss in Stage 2 showed a loss against the repeated baseline.

Table 9 Stage 3 robustness (n = 24 projects): repeated within-project (median over seeds 17, 42, 73) versus LOPO ranking metrics. Two-sided paired Wilcoxon signed-rank tests with Holm adjustment across the nine tests. Recall@K uses K equal to the number of flaky tests in the evaluated set.

Metric	Model	Within (median)	LOPO (median)	Median diff.	Positive projects	W	Holm p
PR-AUC	LR	0.345	0.089	0.256817	24/24	0	1.1×10^{-6}
PR-AUC	RF	0.893	0.025	0.810365	24/24	0	1.1×10^{-6}
PR-AUC	HGB	0.886	0.035	0.781108	24/24	0	1.1×10^{-6}
Recall@K	LR	0.330	0.013	0.230000	18/24	3	4.3×10^{-4}
Recall@K	RF	0.833	0.000	0.750000	24/24	0	1.1×10^{-4}
Recall@K	HGB	0.800	0.000	0.732143	24/24	0	1.1×10^{-4}
Recall@1%	LR	0.183	0.000	0.088043	14/24	10	1.6×10^{-3}
Recall@1%	RF	0.500	0.000	0.500000	24/24	0	1.1×10^{-4}
Recall@1%	HGB	0.500	0.000	0.500000	24/24	0	1.1×10^{-4}

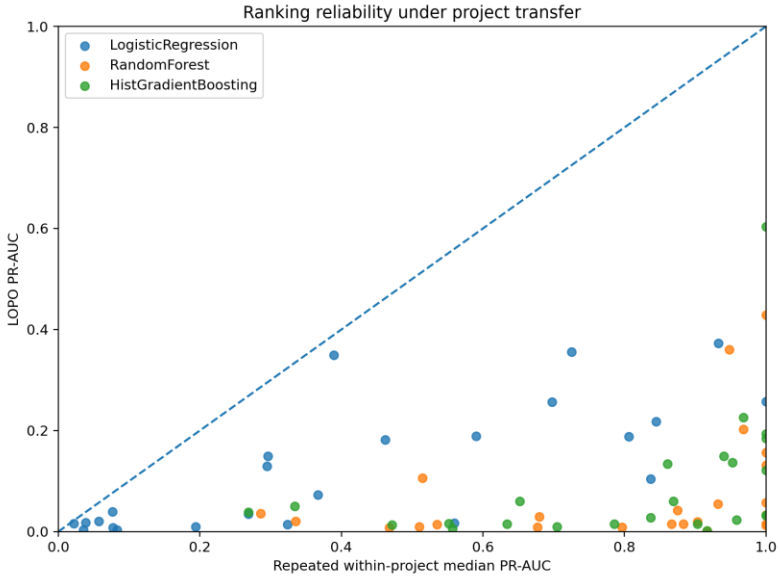


Fig. 6 Stage 3 robustness (n = 24 projects): repeated within-project median PR-AUC (x-axis) against LOPO PR-AUC (y-axis), one point per project and classifier. Points below the dashed diagonal indicate ranking loss under LOPO.

Positive-count sensitivity. Many targets contain few flaky tests, which makes their PR-AUC noisy. Restricting the analysis to the nine targets with at least 30 flaky tests left the conclusion unchanged: the median PR-AUC loss was 0.369636 for LR, 0.587491 for RF, and 0.742086 for HGB, and it was positive in all 9 targets for every classifier (Table 10).

Prevalence strata. The ranking loss was positive in every project of both prevalence strata for every classifier (Table 11). For RF and HGB the median losses were large in both strata (RF 0.728265 and 0.837798; HGB 0.657119 and 0.807746 for the low- and high-prevalence strata). For LR the loss was much smaller among low-prevalence targets (0.074321) than among high-prevalence targets (0.421153), consistent with the descriptive correlation between prevalence and LR's PR-AUC loss reported in Section 5.5.

Table 10 Stage 3 frozen robustness criteria. A: repeated-within PR-AUC loss ≥ 0.05 with Holm $p < 0.05$ (n = 24). B: median PR-AUC loss ≥ 0.05 among targets with ≥ 30 flaky tests (n = 9). C: median PR-AUC loss > 0

Model	A: median loss	A: Holm p	A	B: median loss (≥ 30)	B	C: low prev.	C: high prev.	C	All
LR	0.256817	1.1×10^{-6}	Pass	0.369636	Pass	0.074321	0.421153	Pass	Pass
RF	0.810365	1.1×10^{-6}	Pass	0.587491	Pass	0.728265	0.837798	Pass	Pass
HGB	0.781108	1.1×10^{-6}	Pass	0.742086	Pass	0.657119	0.807746	Pass	Pass

Stage 3 verdict: ROBUST_RANKING_COLLAPSE (3/3 models passed all criteria).

Table 11 Stage 3 prevalence-stratified ranking loss (median split at 1.94% target prevalence). "Positive fraction" is the fraction of projects in the stratum with a PR-AUC loss > 0.

Model	Stratum	Projects	Median PR-AUC loss	Median Recall@K loss	Positive fraction
LR	Low	12	0.074321	0.079670	1.00
LR	High	12	0.421153	0.487179	1.00
RF	Low	12	0.728265	0.656481	1.00
RF	High	12	0.837798	0.786111	1.00
HGB	Low	12	0.657119	0.633333	1.00
HGB	High	12	0.807746	0.833333	1.00

Answer to RQ4. The ranking collapse is robust. All three classifiers passed all three frozen Stage 3 criteria: the PR-AUC loss persisted against repeated within-project baselines (24 of 24 projects for every classifier), among targets with at least 30 flaky tests (9 of 9), and in both prevalence strata (every project). The verdict was ROBUST_RANKING_COLLAPSE with 3 of 3 classifiers passing. These analyses reuse the Stage 2 projects and therefore strengthen, but do not independently replicate, the Stage 2 result.

5.5 RQ5: Operational ranking, calibration, and static feature shift

Operational ranking. Review-budget metrics describe what a developer inspecting the top of the ranked list would find (Table 9 and Fig. 7). With K equal to the number of truly flaky tests in the evaluated set, the median within-project Recall@K was 0.330 for LR, 0.833 for RF, and 0.800 for HGB, and the median LOPO Recall@K was 0.013, 0.000, and 0.000. The median loss was 0.230000 (LR; positive in 18 of 24 projects), 0.750000 (RF; 24 of 24), and 0.732143 (HGB; 24 of 24), all significant after Holm adjustment. Under LOPO, the top-K list contained no flaky test at all in 12 (LR), 17 (RF), and 14 (HGB) of the 24 targets. Recall@1% showed the same pattern: the median within-project value was 0.183, 0.500, and 0.500, and the median LOPO value was 0.000 for all three classifiers. The median loss was 0.088043 for LR (14 of 24 projects; Holm-adjusted $p = 1.6 \times 10^{-3}$) and 0.500000 for both RF and HGB (24 of 24; Holm-adjusted $p = 1.1 \times 10^{-4}$), and the top 1% of the LOPO ranking contained no flaky test in 16, 18, and 18 targets. A team that inspects the highest-ranked tests in a new project would, in most targets, find none of its flaky tests there.

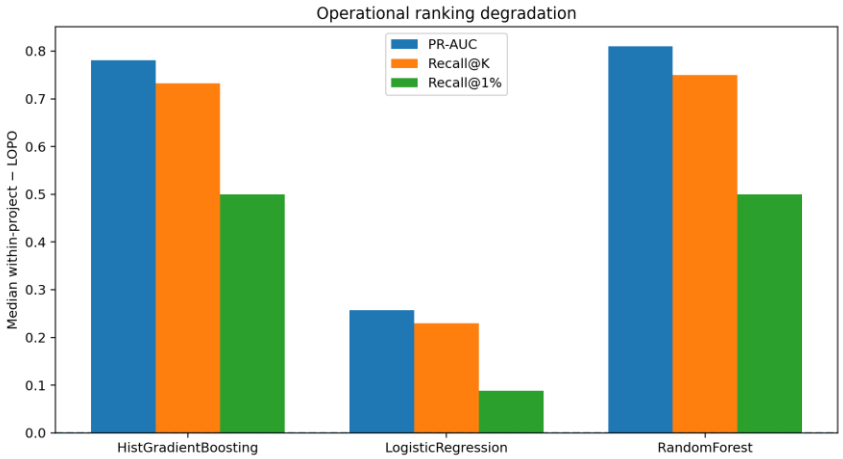


Fig. 7 Stage 3 robustness: median project-level loss (repeated within-project minus LOPO) in PR-AUC, Recall@K, and Recall@1%, by classifier (n = 24 projects).

Calibration. Calibration worsened materially under LOPO for LR and HGB (Table 12). LR's median Brier score rose from 0.100819 to 0.217697 and its median ECE-10 from 0.188281 to 0.398315; HGB's median Brier score rose from 0.007334 to 0.072793 and its median ECE-10 from 0.009962 to 0.165335. The median mean predicted probability on LOPO targets was 0.430 for LR and 0.184 for HGB, against a median target prevalence of 0.019, so both classifiers over-predicted flakiness on unseen projects, which is consistent with the class weighting used in training. RF remained comparatively well calibrated in the median (Brier 0.007707 to 0.020432; ECE-10 0.012765 to 0.015523; median mean probability 0.018) even though its ranking collapsed most severely. Calibration in this sense therefore does not account for the ranking collapse: the model whose average probabilities matched target prevalence best was also the model whose ranking was, at the median, no better than chance.

Table 12 Stage 3 calibration diagnostics (medians over n = 24 projects). ECE-10 is the expected calibration error over ten equal-width probability bins.

Model	Brier within	Brier LOPO	ECE-10 within	ECE-10 LOPO	Mean LOPO probability	Target prevalence
LR	0.100819	0.217697	0.188281	0.398315	0.430	0.019
RF	0.007707	0.020432	0.012765	0.015523	0.018	0.019
HGB	0.007334	0.072793	0.009962	0.165335	0.184	0.019

Static feature shift. The median absolute standardised mean difference between source and target, taken over the 25 predictors, ranged from 0.090 to 0.458 across targets (median 0.204). Its association with the PR-AUC loss was weak and not significant for any classifier (Spearman ρ = 0.351, p = 0.092 for LR; ρ = 0.150, p = 0.485 for RF; ρ = 0.016, p = 0.942 for HGB; Table 13). The Stage 2 analysis, which used the single-split baseline, gave similar values (ρ = 0.203, 0.133, and 0.032). Larger static mean shift did not go with lower LOPO PR-AUC either; the descriptive correlations between shift and LOPO PR-AUC were positive (ρ = 0.473 for LR and 0.463 for HGB), which suggests that shift magnitude is entangled with other project characteristics rather than acting as a simple cause. Target prevalence was strongly associated with LR's PR-AUC loss (ρ

= 0.663, $p = 4.2 \times 10^{-4}$) but not with the losses of RF ($\rho = -0.043$) or HGB ($\rho = 0.048$). These correlations are descriptive, unadjusted, and computed on 24 projects; they indicate model-specific heterogeneity and do not identify a general cause of the ranking collapse.

Table 13 Stage 3 descriptive Spearman correlations across the 24 targets (unadjusted p-values). "Static shift" is the median absolute standardised mean difference over the 25 predictors.

Association	LR ρ	LR p	RF ρ	RF p	HGB ρ	HGB p
Static shift vs PR-AUC loss	0.351	0.092	0.150	0.485	0.016	0.942
Prevalence vs PR-AUC loss	0.663	0.000	−0.043	0.840	0.048	0.824
Flaky count vs PR-AUC loss	0.435	0.034	−0.326	0.120	−0.271	0.200
Static shift vs LOPO PR-AUC	0.473	0.020	0.312	0.138	0.463	0.023

Answer to RQ5. In operational terms the collapse is stark: under LOPO the top-K and top-1% of the ranking contained no flaky test in most targets, against within-project Recall@K of about 0.8 for RF and HGB. Calibration worsened materially for LR and HGB but not for RF, so it does not explain the ranking collapse. Static feature-mean shift was not associated with the size of the loss, and prevalence was associated with the loss only for LR.

6 Discussion

6.1 Strong within-project scores did not certify cross-project rankings

The central finding is that, for the static predictors and classifiers studied, within-project performance was a poor guide to cross-project performance. RF and HGB separated flaky from non-flaky tests almost perfectly within projects, with median ROC-AUC above 0.99 and median PR-AUC around 0.9, yet on unseen projects their median ROC-AUC was 0.49 and 0.58. A within-project score answers whether flaky tests in a project resemble other flaky tests of the same project in the space of static metrics. A cross-project deployment asks whether flaky tests in one project resemble flaky tests in other projects. The results indicate that, for this predictor set, the first property held strongly and the second hardly at all. The models most able to fit project-specific structure, RF and HGB, showed the highest within-project scores and the largest losses; LR, which can represent only a linear combination of the predictors, gained less within projects and lost less across them.

The study does not establish why the within-project relationships failed to transfer. One plausible contributor is that within-project splits keep different tests of the same project, which share code base, conventions, and in many cases class-level metric values, on both sides of the split, so that a flexible model can identify the regions of feature space in which a project's flaky tests are concentrated without learning anything that generalises to other projects. The within-project split was grouped only by the testCase identifier, which is nearly unique, so it did not prevent such sharing. This explanation is consistent with the pattern across classifiers, but it was not tested directly, and Section 8 lists it as a threat to the interpretation of within-project scores.

6.2 Why threshold adaptation could not help

Threshold adaptation is the natural deployment-time response to a model whose scores are shifted on a new project, and it addressed a real problem. At the default threshold, RF flagged no test at all in 23 of 24 targets, a failure mode that would be immediately visible to a team deploying it. Choosing the threshold from project-grouped out-of-fold predictions on the source projects removed this failure in every case, without using any target label. The resulting predictions, however, were barely better than the zero-positive output they

replaced, because a threshold can only cut an existing ranking, and the ranking had already collapsed. The behaviour of LR illustrates the same point from the opposite direction: the source data suggested raising LR's threshold, which reduced the number of flagged tests by a factor of about six but did not improve target F1, because the tests that LR scored highest on the target were rarely the flaky ones.

The project-grouped out-of-fold analysis on the source projects is informative in its own right. The best F1 attainable on those predictions was between 0.03 and 0.05 at the median for every classifier, which already indicated that the classifiers did not rank flaky tests well on projects they had not seen. This diagnostic requires no target labels and could be computed before deployment. It does not replace a target evaluation, but it gives a cheap warning that a model's within-project score should not be expected to carry over.

6.3 Relation to the original hypothesis and to prior cross-project work

The original hypothesis concerned the fragility of predictions to the removal of particular feature families. It failed under its frozen rule, and the paper reports it as not supported. With hindsight, the LOPO baseline against which fragility was measured was already close to zero for RF and HGB, which left little room for further degradation; the feature-family question may be better posed within projects or for models whose cross-project ranking is intact. TEST_SIZE_COMPLEXITY was the family whose removal most often caused the largest F1 drop, but no claim is made about it beyond that descriptive observation.

The results do not contradict Afeltra et al. (2024). That study reported that cross-project flaky-test prediction is challenging, which is consistent with the present findings, and that filtering and transfer-learning approaches, in particular TrAdaBoost, improve performance. The present study evaluated plain source-trained classifiers with no filtering, no transfer learning, and no target labels, and asked a different question: how much within-project performance survives strict deployment, and whether the loss lies in the ranking or in the threshold. The decomposition suggests a criterion for judging cross-project methods. A method that improves cross-project F1 by changing the effective decision threshold is addressing the smaller of the two components observed here; a method that restores PR-AUC or Recall@K on unseen projects is addressing the larger. Reporting threshold-free ranking metrics alongside F1 would make this distinction visible.

6.4 What calibration and shift diagnostics add

Calibration and ranking are distinct properties, and the RF results show that they can diverge sharply. RF's average predicted probability on unseen projects was close to the target prevalence, and its Brier score and ECE-10 deteriorated only modestly, yet its ranking was, at the median, no better than chance. For a rare positive class, a model that assigns uniformly low probabilities can be well calibrated on average while being useless for ranking. Calibration metrics should therefore not be read as evidence of cross-project reliability on their own. Conversely, LR and HGB became markedly miscalibrated under LOPO, over-predicting flakiness, which matters for any use of their probabilities as risk estimates.

The weak and non-significant correlations between static mean shift and PR-AUC loss indicate that a simple summary of distributional distance between source and target does not predict how badly a target will be affected. This does not mean that distribution shift is irrelevant; the collapse itself is a manifestation of shift. It suggests rather that the relevant shift concerns the relationship between static metrics and flakiness, which varies across projects, more than the marginal distributions of the metrics, which the standardised mean difference measures (Moreno-Torres et al. 2012). The strong association between prevalence and LR's loss, absent for RF and HGB, is best treated as model-specific heterogeneity that a larger sample of projects would be needed to explain.

7 Practical Implications

For practitioners who consider adopting a static flaky-test predictor trained on other projects, the results suggest four precautions. First, a model's within-project or pooled performance should not be taken as an estimate of its performance on a new project; strict project-level holdout is the relevant evaluation. Second, a model that flags nothing at the default threshold on a new project is exhibiting an easily repaired symptom of a deeper problem: re-selecting the threshold on source data removes the symptom but should not be expected to produce useful predictions. Third, project-grouped out-of-fold evaluation on the training projects is an inexpensive, label-free warning sign; if a model cannot rank flaky tests on held-out source projects, it is unlikely to do so on the target. Fourth, when the predictor is used to prioritise tests for inspection or rerunning, Recall@K or a fixed review budget such as Recall@1% describes its usefulness more directly than F1 or accuracy.

For researchers, the results support reporting threshold-free ranking metrics alongside threshold-dependent ones in cross-project flaky-test studies, together with the number of targets that receive no positive prediction, the procedure used to select thresholds, and confirmation that no target labels were used in that procedure. Evaluating new cross-project methods by whether they restore target PR-AUC and Recall@K, and not only F1, would show whether they repair the ranking or only the threshold. Separating discovery from confirmation, as done here, also makes it possible to report a discovery-driven redesign without overstating its evidential weight.

8 Threats to Validity

Construct validity. The flaky labels are those of the public benchmark and inherit its labelling procedure and limitations; tests that are flaky but were never observed to flake would be labelled non-flaky. The 25 static metrics cannot capture every cause of flakiness, many of which, such as concurrency, timing, and test-order dependence, are dynamic in nature (Luo et al. 2014). The findings therefore concern these predictors and these classifiers, not flaky-test detection in general. PR-AUC, Recall@K, and Recall@1% are analytical proxies for operational use. Recall@K sets K to the true number of flaky tests in the evaluated set, which is unknown in practice, and Recall@1% corresponds to very different absolute numbers of inspected tests in a small within-project test set and in a full target project.

Internal validity. Within-project evaluation used a grouped 80/20 split. Stage 2 used a single split, and Stage 3 repeated it with three fixed seeds, which is not exhaustive repeated cross-validation. The grouping key, testCase, is nearly unique, so tests that share class-level metric values could fall on both sides of a within-project split; this may inflate within-project scores and therefore the measured transfer losses (Section 6.1), and it was not controlled. No hyperparameter optimisation was performed, and class weighting and fixed configurations were used; tuned or differently regularised models might transfer differently. The threshold search used a quantile grid and an F1 (or MCC or balanced-accuracy) objective with ties broken toward the higher threshold, and other grids or objectives could select different thresholds. The study was redesigned after the discovery stage. To prevent the redesign from biasing confirmation, all five discovery projects were excluded from both training and targets in Stage 2, and the Stage 2 rules were frozen before any replication outcome was inspected. The Stage 1 ablation used LOPO among only five projects, which limits what the failed fragility test can show.

Conclusion validity. The project is the inferential unit, and the replication sample is 24 projects. Two-sided Wilcoxon signed-rank tests with Holm adjustment were used within Stage 2 and within Stage 3; very small p-values reflect the consistency of the direction of the effects across projects rather than precise effect sizes, and no confidence intervals were computed. Several projects contain few flaky tests, and within-project test sets contained between 3 and 12 flaky tests, which makes individual within-project estimates noisy. The positive-count sensitivity analysis (at least 30

flaky tests, n = 9) and the repeated within-project baseline address this concern but cannot remove it. Stage 3 reuses the Stage 2 projects and is a robustness analysis, not an independent replication. The Spearman correlations are descriptive and unadjusted. Severe class imbalance makes accuracy inappropriate, and it is not used to support any conclusion.

External validity. The dataset comprises 29 open-source projects, and the study used static metrics only. The results may not generalise to other programming languages and ecosystems, to proprietary or cross-company settings, or to other labelling procedures. They do not establish how neural, language-model-based, dynamic, or history-based flaky-test predictors, or transfer-learning and filtering methods, behave under the same deployment conditions. The audit design itself, separating ranking from threshold loss under strict project holdout, can be applied to such predictors.

9 Data and Code Availability

The dataset is publicly available in the replication repository of Afeltra et al. (2024) (https://github.com/alfcan/crossproject-flaky-test-prediction). This study used commit 8fe244f03122706763484dea7b4d98b09fefb923, at which the SHA-256 digest of dataset.csv is 6c2066eb1bf1824887a72c6f6df3cb5106894efde206686eb564287fe6547e50. The study comprises four executed notebooks, which reproduce the Stage 0 audit, the Stage 1 discovery analysis, the Stage 2 independent replication, and the Stage 3 robustness and mechanism analysis, together with the frozen protocol of each modelling stage. Each notebook re-clones the repository, verifies the commit and the digest, reproduces the Stage 0 counts, and writes integrity manifests, metric tables, prediction files, figures, and its verdict. The executed notebooks and their outputs have been retained, and all numerical results reported in this paper were taken from these outputs.

10 Conclusion

This paper reported a staged reliability audit of static flaky-test prediction under cross-project deployment, using the public 29-project benchmark. An initial feature-fragility hypothesis failed in discovery, but the discovery data revealed a collapse from within-project to cross-project performance. Tested on 24 held-out projects under frozen rules, that collapse replicated: cross-project flaky-class F1 fell by a median of 0.800 for random forest and 0.720 for histogram gradient boosting, and by a smaller, statistically significant 0.093 for logistic regression, which did not meet the frozen effect criterion. The loss was dominated by ranking collapse, with median PR-AUC losses of 0.264 to 0.839, and median cross-project ROC-AUC close to 0.5. Decision thresholds selected from source projects alone removed zero-positive output but recovered little target F1. The ranking collapse persisted under repeated within-project baselines, among targets with many flaky tests, across prevalence strata, and in review-budget metrics; calibration and static feature shift did not explain it. For these predictors and this benchmark, strong within-project scores did not indicate usable cross-project rankings, and threshold adjustment could not substitute for a ranking that transfers. Applying the same decomposition to dynamic, history-based, and language-model-based predictors, and to transfer-learning methods, is a natural next step.

Ethics approval. This study analysed a publicly available software-engineering dataset of test-case metrics from open-source projects and did not involve human participants.

Data and code availability. See Section 9.

Use of generative AI. Generative AI tools were used to support language refinement, manuscript organization, and drafting assistance. The research questions, experimental design, code execution, data analysis, result verification, interpretation, and final scholarly responsibility remained with the author. All numerical results reported in the manuscript were derived from executed experimental pipelines and checked against retained output files.

REFERENCES

Afeltra A, Cannavale A, Pecorelli F, Pontillo V, Palomba F (2024) A large-scale empirical investigation into cross-project flaky test prediction. IEEE Access 12:131255–131265. https://doi.org/10.1109/ACCESS.2024.3458184

Alshammari A, Morris C, Hilton M, Bell J (2021) FlakeFlagger: Predicting flakiness without rerunning tests. In: Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE 2021), pp 1572–1584

Austin PC (2009) Balance diagnostics for comparing the distribution of baseline covariates between treatment groups in propensity-score matched samples. Statistics in Medicine 28(25):3083–3107

Bell J, Legunsen O, Hilton M, Eloussi L, Yung T, Marinov D (2018) DeFlaker: Automatically detecting flaky tests. In: Proceedings of the 40th International Conference on Software Engineering (ICSE 2018), pp 433–444

Breiman L (2001) Random forests. Machine Learning 45(1):5–32. https://doi.org/10.1023/A:1010933404324

Brier GW (1950) Verification of forecasts expressed in terms of probability. Monthly Weather Review 78(1):1–3

Camara B, Silva M, Endo A, Vergilio S (2021) On the use of test smells for prediction of flaky tests. In: Proceedings of the 6th Brazilian Symposium on Systematic and Automated Software Testing (SAST 2021), pp 46–54

Chicco D, Jurman G (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21:6. https://doi.org/10.1186/s12864-019-6413-7

Dai W, Yang Q, Xue G-R, Yu Y (2007) Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning (ICML 2007), pp 193–200. https://doi.org/10.1145/1273496.1273521

Eck M, Palomba F, Castelluccio M, Bacchelli A (2019) Understanding flaky tests: The developer's perspective. In: Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019), pp 830–840

Fatima S, Ghaleb TA, Briand L (2023) Flakify: A black-box, language model-based predictor for flaky tests. IEEE Transactions on Software Engineering 49(4):1912–1927. https://doi.org/10.1109/TSE.2022.3201209

Friedman JH (2001) Greedy function approximation: A gradient boosting machine. Annals of Statistics 29(5):1189–1232. https://doi.org/10.1214/aos/1013203451

Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML 2017), PMLR 70, pp 1321–1330

Haben G, Habchi S, Papadakis M, Cordy M, Le Traon Y (2021) A replication study on the usability of code vocabulary in predicting flaky tests. In: Proceedings of the 18th IEEE/ACM International Conference on Mining Software Repositories (MSR 2021), pp 219–229

Herbold S, Trautsch A, Grabowski J (2018) A comparative study to benchmark cross-project defect prediction approaches. IEEE Transactions on Software Engineering 44(9):811–833. https://doi.org/10.1109/TSE.2017.2724538

Holm S (1979) A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2):65–70

Hosseini S, Turhan B, Gunarathna D (2019) A systematic literature review and meta-analysis on cross project defect prediction. IEEE Transactions on Software Engineering 45(2):111–147. https://doi.org/10.1109/TSE.2017.2770124

Kapoor S, Narayanan A (2023) Leakage and the reproducibility crisis in machine-learning-based science. Patterns 4(9):100804. https://doi.org/10.1016/j.patter.2023.100804

Luo Q, Hariri F, Eloussi L, Marinov D (2014) An empirical analysis of flaky tests. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014), pp 643–653

Moreno-Torres JG, Raeder T, Alaiz-Rodríguez R, Chawla NV, Herrera F (2012) A unifying view on dataset shift in classification. Pattern Recognition 45(1):521–530

Naeini MP, Cooper GF, Hauskrecht M (2015) Obtaining well calibrated probabilities using Bayesian binning. In: Proceedings of the AAAI Conference on Artificial Intelligence 29(1). https://doi.org/10.1609/aaai.v29i1.9602

Parry O, Kapfhammer GM, Hilton M, McMinn P (2022) A survey of flaky tests. ACM Transactions on Software Engineering and Methodology 31(1):17. https://doi.org/10.1145/3476105

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830

Pinto G, Miranda B, Dissanayake S, d'Amorim M, Treude C, Bertolino A (2020) What is the vocabulary of flaky tests? In: Proceedings of the 17th International Conference on Mining Software Repositories (MSR 2020), pp 492–502. https://doi.org/10.1145/3379597.3387482

Pontillo V (2022) Static test flakiness prediction. In: Proceedings of the 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion 2022), pp 325–327

Pontillo V, Palomba F, Ferrucci F (2021) Toward static test flakiness prediction: A feasibility study. In: Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution (MaLTeSQuE 2021), pp 19–24. https://doi.org/10.1145/3472674.3473981

Pontillo V, Palomba F, Ferrucci F (2022) Static test flakiness prediction: How far can we go? Empirical Software Engineering 27(7):187. https://doi.org/10.1007/s10664-022-10227-1

Saerens M, Latinne P, Decaestecker C (2002) Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Computation 14:21–41. https://doi.org/10.1162/089976602753284446

Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE 10(3):e0118432. https://doi.org/10.1371/journal.pone.0118432

Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2017) An empirical comparison of model validation techniques for defect prediction models. IEEE Transactions on Software Engineering 43(1):1–18. https://doi.org/10.1109/TSE.2016.2584050

Turhan B, Menzies T, Bener AB, Di Stefano J (2009) On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering 14(5):540–578

Verdecchia R, Cruciani E, Miranda B, Bertolino A (2021) Know you neighbor: Fast static prediction of test flakiness. IEEE Access 9:76119–76134. https://doi.org/10.1109/ACCESS.2021.3082424

Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bulletin 1(6):80–83. https://doi.org/10.2307/3001968

Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. In: Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2009), pp 91–100. https://doi.org/10.1145/1595696.1595713.

Appendix A Project-level results

Tables A1 and A2 report the Stage 2 and Stage 3 project-level values from which the medians and tests in Section 5 were computed. "Within" in Table A1 is the single Stage 2 within-project split; "Rep. within" in Table A2 is the Stage 3 median over three seeds.

Table A1 Stage 2 independent replication: flaky-class F1 within project, under LOPO at threshold 0.50 ("Default"), and under LOPO at the source-only threshold ("Source"), by project and classifier.

Project	LR within	LR default	LR source	RF within	RF default	RF source	HGB within	HGB default	HGB source
Achilles	0.435	0.000	0.000	1.000	0.000	0.071	1.000	0.000	0.131
Activiti	0.308	0.000	0.000	0.800	0.000	0.049	0.800	0.052	0.085
admiral	0.100	0.075	0.042	0.222	0.000	0.077	0.182	0.038	0.067
adyen-java-api-library	0.700	0.044	0.000	0.875	0.000	0.359	0.800	0.033	0.000
aem-core-wcm-components	0.170	0.083	0.074	0.667	0.000	0.000	0.667	0.069	0.037
Chronicle-Wire	0.231	0.198	0.208	0.375	0.000	0.128	0.308	0.104	0.168
commons-lang	0.081	0.057	0.030	0.500	0.000	0.006	0.333	0.053	0.037
DataflowTemplates	0.467	0.258	0.023	0.875	0.000	0.000	0.875	0.000	0.048
druid	0.021	0.005	0.009	0.222	0.000	0.012	0.300	0.007	0.009
fastjson	0.095	0.019	0.000	0.667	0.000	0.018	0.500	0.014	0.008
flink	0.048	0.015	0.038	0.500	0.000	0.009	0.545	0.036	0.032
graylog2-server	0.106	0.013	0.000	0.909	0.000	0.012	0.833	0.023	0.018
http-request	0.750	0.267	0.276	1.000	0.000	0.000	1.000	0.000	0.295
jackrabbit-oak	0.032	0.006	0.007	0.800	0.000	0.003	0.909	0.003	0.003
Java-WebSocket	0.909	0.030	0.038	0.833	0.000	0.000	0.833	0.000	0.026
mockserver	0.146	0.011	0.000	0.615	0.000	0.076	0.571	0.014	0.016

Project	LR within	LR default	LR source	RF within	RF default	RF source	HGB within	HGB default	HGB source
nacos	0.158	0.065	0.105	0.833	0.000	0.027	0.727	0.014	0.021
ozone	0.043	0.036	0.029	0.000	0.000	0.026	0.800	0.065	0.038
servicecomb-java-chassis	0.615	0.063	0.073	0.889	0.000	0.000	0.941	0.000	0.000
typescript-generator	0.778	0.340	0.000	0.800	0.000	0.593	1.000	0.000	0.213
visualee	0.769	0.350	0.000	0.952	0.000	0.462	0.870	0.167	0.568
vpc-java-sdk	0.093	0.254	0.259	0.909	0.000	0.000	0.727	0.000	0.000
wildfly	0.435	0.231	0.463	0.714	0.000	0.022	0.667	0.225	0.170
wro4j	0.036	0.027	0.000	0.857	0.000	0.061	0.667	0.097	0.097

Table A2 PR-AUC by project and classifier: Stage 2 single-split within-project value, Stage 3 repeated within-project median, and LOPO value.

Project	LR within	LR rep. within	LR LOPO	RF within	RF rep. within	RF LOPO	HGB within	HGB rep. within	HGB LOPO
Achilles	0.712	0.698	0.257	1.000	1.000	0.157	1.000	1.000	0.121
Activiti	0.560	0.560	0.017	1.000	1.000	0.033	1.000	1.000	0.033
admiral	0.077	0.077	0.040	0.286	0.286	0.036	0.269	0.269	0.038
adyen-java-api-library	0.648	0.806	0.188	0.968	0.968	0.202	0.940	0.940	0.149
aem-core-wcm-components	0.367	0.367	0.073	0.854	0.903	0.019	0.875	0.958	0.023
Chronicle-Wire	0.120	0.296	0.149	0.249	0.515	0.106	0.454	0.861	0.135
commons-lang	0.269	0.269	0.035	0.656	0.677	0.009	0.652	0.652	0.060
DataflowTemplates	0.501	0.462	0.181	0.943	0.875	0.042	0.918	0.869	0.060
druid	0.097	0.083	0.003	0.441	0.467	0.008	0.406	0.557	0.006
fastjson	0.248	0.324	0.014	0.550	0.535	0.015	0.444	0.472	0.014
flink	0.136	0.057	0.020	0.510	0.510	0.010	0.578	0.552	0.016
graylog2-server	0.230	0.077	0.008	0.858	0.797	0.009	0.809	0.705	0.010
http-request	0.625	0.590	0.189	1.000	1.000	0.131	1.000	1.000	0.193
jackrabbit-oak	0.036	0.036	0.003	0.943	0.917	0.002	0.967	0.917	0.002
Java-WebSocket	1.000	1.000	0.258	1.000	1.000	0.058	1.000	1.000	0.184
mockserver	0.164	0.194	0.010	0.660	0.680	0.029	0.612	0.634	0.015
nacos	0.244	0.295	0.130	0.887	0.883	0.015	0.716	0.786	0.015
ozone	0.319	0.022	0.016	1.000	0.867	0.015	0.943	0.837	0.027

Project	LR within	LR rep. within	LR LOPO	RF within	RF rep. within	RF LOPO	HGB within	HGB rep. within	HGB LOPO
servicecomb-java-chassis	1.000	0.837	0.105	1.000	1.000	0.015	1.000	1.000	0.032
typescript-generator	0.938	0.933	0.373	0.957	0.948	0.361	1.000	0.968	0.226
visualee	0.725	0.725	0.356	1.000	1.000	0.429	0.981	1.000	0.603
vpc-java-sdk	0.424	0.389	0.349	1.000	1.000	0.013	0.903	0.903	0.015
wildfly	0.844	0.844	0.218	0.927	0.932	0.055	0.903	0.953	0.137
wro4j	0.039	0.039	0.018	0.806	0.335	0.020	0.833	0.334	0.051

Appendix B Classifier configurations

Table B1 Fixed classifier configurations used in all stages (scikit-learn). No hyperparameter tuning was performed.

Classifier	Configuration
Logistic regression (LR)	StandardScaler followed by LogisticRegression(class_weight = balanced, solver = liblinear, max_iter = 2000, random_state = 42)
Random forest (RF)	RandomForestClassifier(n_estimators = 200, max_features = sqrt, min_samples_leaf = 2, class_weight = balanced_subsample, random_state = 42)
Histogram gradient boosting (HGB)	HistGradientBoostingClassifier(learning_rate = 0.05, max_iter = 200, max_leaf_nodes = 15, l2_regularization = 1.0, random_state = 42), fitted with balanced sample weights
Within-project split	StratifiedGroupKFold(n_splits = 5, shuffle = True) grouped by testCase; first fold as test set; random state 42 in Stage 2 and 17, 42, 73 in Stage 3
Source-only threshold	GroupKFold(n_splits = 5) by project over the 23 source projects; candidates 0.50 and 500 quantiles (0.001–0.999) of out-of-fold probabilities; ties broken toward the higher threshold