

---

**Volume 1 Issue 1**

**Article Number: JASAT20260005**

---

**Early Prediction of GitHub Issue Escalation Using Text and Collaboration Features**

I. Wayan Budi Sentana<sup>1,2</sup> and Gurdip Kaur Saminder Singh<sup>\* 3</sup>

<sup>1</sup>School of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, NSW 2109, Australia

<sup>2</sup>Department of Information Technology, Politeknik Negeri Bali, Jimbaran, Badung, Bali 80364, Indonesia

<sup>3</sup>Open University Malaysia (OUM), Kuala Lumpur, Malaysia

---

**Abstract**

*Problem:* GitHub issues can develop into prolonged, contentious, or coordination-intensive episodes, but escalation is usually recognized only after substantial maintainer effort has been spent. *Motivation:* An early warning could help maintainers prioritize attention while preserving human judgment. *Gap:* Existing work largely predicts assignment, severity, priority, or resolution time; these targets do not jointly represent escalation, and many approaches underuse the interaction process visible at the beginning of an issue. *Framework:* This paper proposes a leakage-aware conceptual framework that predicts escalation at fixed early cutoffs, including issue creation, the first maintainer response, and the first few comments. Observable future events—such as reopening, priority promotion, cross-repository transfer or reference, senior-contributor assignment, abnormal unresolved duration, discussion growth, and validated conflict—define the outcome but are excluded from predictors. Early predictors combine title, body, and comment representations with participant growth, response timing, interaction networks, maintainer involvement, and repository controls. Interpretable linear and tree models and multimodal neural fusion are specified, together with calibrated, alert-budget-aware evaluation. *Contributions:* The proposal provides an operational escalation construct, a temporal feature taxonomy, a reproducible multi-repository methodology, and an explanation layer for maintainer-facing alerts. No empirical performance claim is made; the expected contribution is a testable design for future evaluation and responsible deployment.

---

**Keywords:** GitHub Issues; Issue Escalation; Mining Software Repositories; Natural Language Processing; Collaboration Networks; Early Prediction; Software Maintenance

---

**I. Introduction**

Issue trackers are not merely defect databases: they are public coordination spaces in which reports are interpreted, negotiated, assigned, and linked to work. Large-scale analysis has shown that GitHub issue trackers contain heterogeneous maintenance activity and participation patterns [1]. GitHub's visibility also exposes social cues and work histories that affect collaboration [2]. These affordances create useful signals, but repository mining is vulnerable to incomplete

---

<sup>\*</sup>**Corresponding Author:** Gurdip Kaur Saminder Singh ([drguripsaini@gmail.com](mailto:drguripsaini@gmail.com))

**Received:** 25 June 2026; **Revised:** 29 July 2026; **Accepted:** 31 August 2026; **Published:** 30 September 2026

© 2026 The Author(s).

This is an open access article licensed under a [Creative Commons Attribution-NonCommercial 4.0 International Licence \(CC BY-NC 4.0\)](https://creativecommons.org/licenses/by-nc/4.0/).

histories, non-development activity, and misleading units of analysis [3]. Discussion itself matters when contributors and maintainers evaluate proposed work [4].

Some issues become unusually costly: they remain unresolved relative to repository norms, attract rapidly expanding or tense discussion, require repeated maintainer intervention, are reopened, move to a higher priority, or are redirected across repository boundaries. We call this observable trajectory *issue escalation*. Escalation is broader than severity and distinct from a long lifetime: a technically severe report may be handled calmly, while an apparently routine request may become coordination-intensive. Early identification is difficult because decisive outcome events occur later and because initial wording can be sparse, ambiguous, or strategically neutral.

Text alone therefore offers an incomplete view. Titles, descriptions, and early comments express urgency, uncertainty, severity language, and affect, but the emerging interaction process provides complementary evidence: response latency, participant growth, reciprocity, repeated clarification, maintainer workload, and prior ties. A responsible predictor must also enforce a strict temporal boundary; otherwise later labels, assignments, and discussion outcomes silently leak the target into the features.

This paper proposes, but does not empirically validate, a framework for early escalation prediction. Its expected practical value is risk-ranked triage: identifying discussions that warrant a timely human review, not automatically sanctioning contributors or closing issues. The contributions are:

1. an observable, repository-adaptable escalation construct that separates post-cutoff evidence from pre-cutoff predictors;
2. a temporal taxonomy combining text, collaboration, and repository-context features at several actionable early windows;
3. a reproducible evaluation design with temporal and cross-repository tests, imbalance-aware metrics, ablations, and leakage controls; and
4. an explanation and alert design that exposes risk factors, uncertainty, and model limitations to maintainers.

## II. Background and Related Work

### A. GitHub Issue Management and Triage

Classic triage research treats routing as a recommendation problem. Anvik *et al.* inferred suitable bug fixers from historical reports and assignments [5]; Jeong *et al.* modeled reassignment paths with bug-tossing graphs [6]. These methods inform who should act, whereas escalation prediction asks whether an issue is likely to demand exceptional coordination. GitHub studies further show that issue records should be interpreted together with visible social activity [1, 2]; however, visibility can also encode popularity and prior-status effects.

### B. Bug Severity, Priority, and Resolution-Time Prediction

Lamkanfi *et al.* predicted reported-bug severity from textual descriptions [7], while Giger *et al.* modeled bug-fix time using report and project attributes [8]. Kikas *et al.* showed that dynamic and contextual features can support GitHub issue-lifetime prediction [9]. Izadi *et al.* jointly studied issue objective and priority from report content [10]. Shihab *et al.* characterized and predicted reopened bugs [11]. These targets overlap with escalation indicators, but none alone captures the compound social and maintenance trajectory proposed here. Moreover, a resolution-time feature is a valid future labeling signal but not an early predictor.

### C. Textual and Sentiment Features in Software Engineering

General-purpose sentiment tools can misread software-engineering language, motivating domain validation [12]. Calefato *et al.* developed and assessed a software-development sentiment classifier [13]. Work on politeness associates linguistic style with participation outcomes but does not establish that manners cause technical success [14]. Toxicity has been examined in open-source discussions [15]; locked GitHub issues provide a related but platform-mediated view of heated discussion [16]. Qiu *et al.* studied interpersonal conflict detection across issues and code review [17]. Together, these studies justify affective and pragmatic features, while warning that tool scores are noisy, culturally contingent measurements rather than ground truth.

## D. Collaboration and Social Interaction Features

Developer networks reveal latent project structure [18], yet conclusions depend on how nodes and edges are constructed [19]. Fine-grained network definitions can better align inferred communities with developer roles [20]. On GitHub, both social and technical factors influence contribution evaluation [21]. Escalation modeling should therefore test interaction features explicitly, define event-to-edge mappings, and control for repository and contributor history rather than interpreting centrality as an intrinsic personal trait.

## E. Early-Warning Models and Research Gap

Prior work usually predicts a single terminal property or updates a lifetime estimate as more events arrive. The missing unit is an *early, compound escalation risk* whose outcome is observable after a fixed cutoff and whose predictors include the emerging collaboration process. Event archives such as GHTorrent can support timeline reconstruction [22], while bot detection is necessary because automated comments can distort response rates and participant networks [23]. Table I contrasts representative work. “Early” indicates a prediction explicitly possible before the target event; “partial” collaboration means limited social or dynamic attributes rather than an interaction layer.

Table 1: Comparison of representative related studies.

| Study                | Prediction Target            | Data Source            | Text | Collab. | Early   | Main Limitation for Escalation Research                    |
|----------------------|------------------------------|------------------------|------|---------|---------|------------------------------------------------------------|
| Anvik et al. [5]     | Fixer recommendation         | Bug repositories       | Y    | Partial | Y       | Routes reports; does not predict an escalation trajectory. |
| Jeong et al. [6]     | Reassignment path            | Bug repositories       | Y    | Partial | Y       | Assignment graph is narrower than discussion dynamics.     |
| Lamkanfi et al. [7]  | Bug severity                 | Bug repositories       | Y    | –       | Y       | Severity is a single report property.                      |
| Giger et al. [8]     | Fix time                     | Bug repositories       | Y    | Partial | Y       | Predicts duration, not compound escalation.                |
| Kikas et al. [9]     | Issue lifetime               | GitHub issues          | Y    | Partial | Y       | Dynamic context is used, but escalation is not the target. |
| Izadi et al. [10]    | Objective and priority       | Issue reports          | Y    | –       | Y       | Focuses on report content and issue labels.                |
| Shihab et al. [11]   | Reopened bug                 | OSS bug reports        | Y    | Partial | Y       | Reopening is only one escalation indicator.                |
| Miller et al. [15]   | Toxic discussion content     | OSS discussions        | Y    | Partial | –       | Characterizes toxicity rather than future escalation.      |
| Ferreira et al. [16] | Locked-issue characteristics | GitHub issues          | Y    | Y       | –       | Locking is a retrospective, platform-specific outcome.     |
| Qiu et al. [17]      | Interpersonal conflict       | Issues/code review     | Y    | Y       | Partial | Conflict is one component of escalation.                   |
| Tsay et al. [21]     | Contribution evaluation      | GitHub pull requests   | Y    | Y       | Partial | Target and unit are pull-request decisions.                |
| Proposed framework   | Compound escalation risk     | GitHub issue timelines | Y    | Y       | Y       | Conceptual until evaluated across repositories.            |

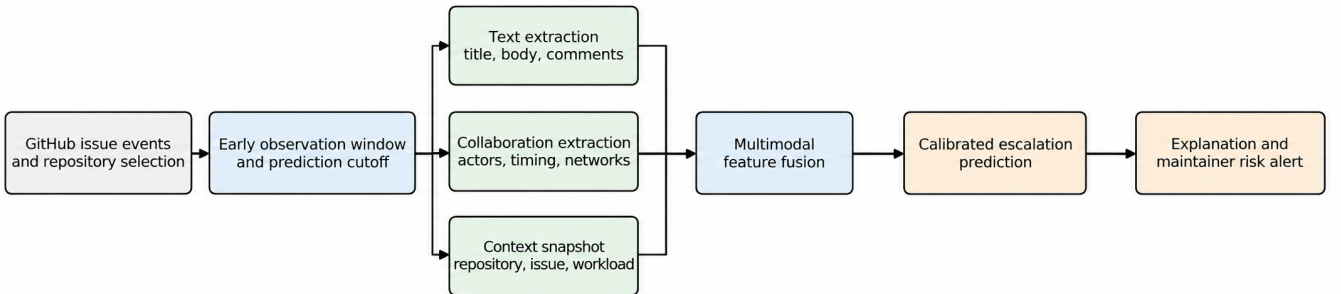


Figure 1: Conceptual framework for early escalation prediction. Every extraction branch receives the same cutoff-time prefix; future events are reserved for label construction.

### III. Problem Definition

Let issue  $i$  be a timestamped sequence  $I_i = (M_i, E_i)$ , where  $M_i$  contains the title, body, creator, initial labels, and repository metadata, and  $E_i = \langle e_{i1}, \dots, e_{in} \rangle$  contains comments, reactions, mentions, assignments, label changes, references, state changes, and transfers. For an observation policy  $w$ , the prediction cutoff  $t_i^p(w)$  may be creation time, elapsed clock time, the first maintainer response, or the  $k$ th eligible comment/event. The early prefix and future suffix are

$$E_i^- = \{e \in E_i : t(e) \leq t_i^p\}, \quad E_i^+ = \{e \in E_i : t(e) > t_i^p\}. \quad (1)$$

The issue representation is

$$\mathcal{I}_i = \{X_i^{text}, X_i^{collab}, X_i^{context}, y_i\}. \quad (2)$$

where the first two feature blocks are computed only from  $(M_i, E_i^-)$ , context is a cutoff-time snapshot, and  $y_i$  is determined from  $E_i^+$  plus eventual issue status.

#### A. Observable Escalation

An escalation-label function  $L_r$  is calibrated for repository or repository family  $r$ :

$$\begin{aligned} y_i &= L_r(\mathbf{s}_i^+, \mathbf{u}_i^+) \in \{0, 1\}, \\ \hat{y}_i &= f_\theta(X_i^{text}, X_i^{collab}, X_i^{context}). \end{aligned} \quad (3)$$

where  $\mathbf{s}_i^+$  records discrete post-cutoff events and  $\mathbf{u}_i^+$  records repository-normalized future trajectories. Observable indicators include reopening; promotion to a label that maintainers validate as high priority, severe, or critical; transfer or cross-repository escalation; assignment to a senior/core contributor; intervention by multiple maintainers; sustained conflict; abnormal discussion growth; and unresolved duration unusually long for comparable issues in  $r$ . Long duration alone is not sufficient where repositories intentionally retain roadmap issues.

The recommended label is an expert-validated ruleset: high-specificity state/priority/transfer events may trigger review directly, whereas noisy indicators such as sentiment, comment growth, or duration require corroboration. Thresholds, label mappings, and combinations are pre-registered from training-period repository norms and annotation, never selected on the test set. An optional ordinal target  $y_i \in \{\text{routine}, \text{emerging}, \text{escalated}\}$  can distinguish warning from realized escalation, provided annotators can apply it reliably.

Figure 2 makes the central separation explicit. A label-changing event after  $t_i^p$  may define  $y_i$  but cannot be a predictor for that prediction. Conversely, an initial label present at creation is eligible as context but should not itself be used to define post-cutoff escalation.

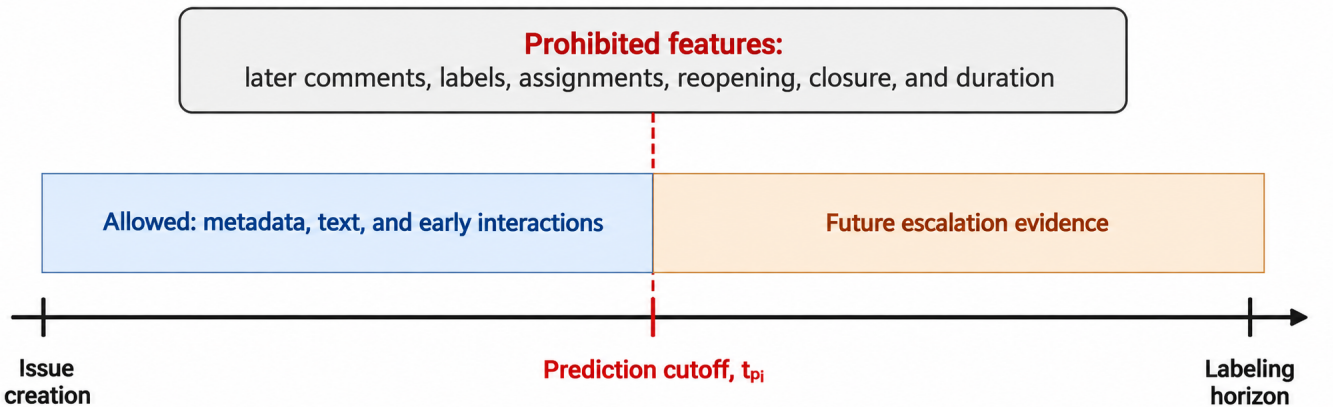


Figure 2: Temporal prediction and leakage prevention. Outcome evidence is measured after the cutoff; predictor extraction stops at the cutoff.

## IV. Proposed Conceptual Framework

Figure 1 presents nine stages: repository selection, window construction, three parallel feature layers, fusion, prediction, explanation, and an alert for human review. The design supports both clock-based and event-based cutoffs and preserves the same cutoff throughout extraction.

### A. Textual Feature Layer

The textual input  $T_i$  concatenates the title, body, and eligible early comments while preserving speaker and time markers. Structural features include length, readability, code-block density, stack-trace signatures, URLs, and issue references. Lexical-pragmatic features cover urgency, uncertainty, severity terminology, questions, requests, repeated clarification, sentiment, emotion, toxicity, and politeness. Affect scores should be obtained from software-engineering-validated models and accompanied by error analysis because technical vocabulary can invert general-language polarity [12, 13].

Representations range from term-frequency-inverse-document-frequency (TF-IDF) baselines to contextual encoders. BERT provides a general bidirectional transformer representation [24]; Sentence-BERT supports efficient semantic similarity [25]; and CodeBERT can represent mixed programming and natural language [26]. A proposed similarity feature compares  $T_i$  only with previously escalated training-period issues. Pretrained encoders must be fine-tuned and evaluated without exposing future or test-period text.

### B. Collaboration Feature Layer

The collaboration prefix  $C_i$  is a temporal, typed interaction graph. Nodes are non-bot participants; directed edges represent replies, mentions, reactions, assignment actions, or references observed by  $t_i^p$ . Features include unique participants, maintainers involved, time to first response, comment-arrival and participant-growth rates, reply count, mention density, reciprocity, actor centrality computed from historical-only graphs, maintainer-to-contributor ratio, prior dyadic interaction, assignee and label changes already observed, reactions, cross-references, clarification cycles, disagreement markers, and response-time irregularity.

The timing qualifier is essential. An assignee change or priority label change before the cutoff is a predictor; the same event after the cutoff is future evidence. Reopening, eventual duration, later intervention, and later discussion growth are never backfilled. Role and network features are recomputed as-of the cutoff to prevent contributor-history leakage.

### C. Context and Control Features

Context controls include repository age and size, contributor count, dominant language, issue type, initial labels, historical issue volume, cutoff-time maintainer workload, and project activity. These variables are necessary because identical interaction rates can mean different things in a small library and a high-volume infrastructure project. Controls should be normalized using training-period repository distributions. Repository identifiers may be used for hierarchical effects, but a deployable cross-project model should not rely on one-hot memorization of project identity.

### D. Feature Fusion and Prediction Layer

Three model families should be compared. Regularized logistic regression provides a sparse, auditable baseline. Random forests or gradient-boosted trees capture nonlinearities and interactions in structured features. A neural multimodal model combines a transformer text encoder with a structured collaboration encoder:

$$h_i^{text} = g_\phi(T_i), \quad (4)$$

$$h_i^{collab} = q_\psi(C_i), \quad (5)$$

$$z_i = [h_i^{text} \| h_i^{collab} \| X_i^{context}], \quad (6)$$

$$P(y_i = 1 \mid \mathcal{I}_i) = \sigma(Wz_i + b). \quad (7)$$

Table 2: Proposed feature taxonomy (relationships are hypotheses, not causal claims).

| Feature Group            | Example Features                                                                    | Extraction Source                 | Availability Window      | Expected Predictive Relationship                                                                     |
|--------------------------|-------------------------------------------------------------------------------------|-----------------------------------|--------------------------|------------------------------------------------------------------------------------------------------|
| Text structure           | Length, readability, code density, stack trace, URLs, references                    | Title/body/eligible comments      | At creation or cutoff    | Detail may reduce ambiguity; unusually dense traces or references may indicate complexity.           |
| Affect and pragmatics    | Sentiment, emotion, toxicity, politeness, urgency, uncertainty, questions, requests | Eligible natural-language spans   | At cutoff                | Urgency/tension may raise risk; respectful clarification may lower or ambiguously mark complex work. |
| Text semantics           | TF-IDF, topic, transformer embedding, similarity to prior escalations               | Cutoff text plus training history | At cutoff                | Semantic resemblance may transfer risk information within comparable domains.                        |
| Participation            | Unique actors, maintainer count/ratio, actor growth, bot-filtered activity          | Comment and event actors          | After interactions begin | Rapid growth or exceptional maintainer involvement may indicate coordination demand.                 |
| Interaction graph        | Replies, mentions, density, reciprocity, historical centrality, prior ties          | Typed cutoff-time network         | At cutoff                | Dense reciprocal exchange may aid resolution or reflect controversy; direction is empirical.         |
| Temporal process         | First-response time, arrival rate, latency variance, clarification cycles           | Event timestamps/order            | Clock/event cutoff       | Bursty or irregular response can signal unresolved coordination.                                     |
| Issue/repository context | Initial labels, issue type, age, size, language, volume, workload, activity         | Cutoff snapshots/history          | Creation or cutoff       | Controls baseline differences in workflow and capacity.                                              |

Here,  $g_\phi$  encodes text with parameters  $\phi$ ;  $q_\psi$  encodes structured or graph features with parameters  $\psi$ ;  $\parallel$  denotes concatenation;  $W, b$  are classifier parameters; and  $\sigma$  is the logistic function. An optional modular score is

$$R_i = \alpha P_i^{text} + \beta P_i^{collab} + \gamma P_i^{context}, \quad \alpha + \beta + \gamma = 1. \quad (8)$$

The nonnegative weights must be learned or validated empirically; no arbitrary values are proposed. Probability calibration and decision thresholds are learned on validation data for each alert budget.

## E. Explainability Layer

Local feature attribution, such as SHAP [27], can summarize which observed factors moved a prediction, while global importance and partial-dependence diagnostics can reveal unstable shortcuts. For consequential routing, an interpretable model may be preferable to post-hoc explanation of a black box [28]. A maintainer alert should therefore show calibrated risk, the cutoff, top feature groups, missing-data warnings, and comparable historical cases. Attributions describe associations, not causes. A human decides whether to request clarification, add expertise, moderate, or take no action.

## V. Research Questions and Proposed Methodology

The future empirical study is organized around four research questions:

**RQ1:** How accurately can GitHub issue escalation be predicted using information from an early observation window?

**RQ2:** What predictive value is added by collaboration features beyond textual and repository-context features?

**RQ3:** Which textual and collaboration features contribute most strongly to early escalation predictions?

**RQ4:** How robust are escalation models across repositories, project domains, and observation windows?

### A. Data Collection

Issue and event timelines can be reconstructed from a public research archive such as GHTorrent [22], another scholarly described repository dataset, or a reproducible archival capture. Inclusion criteria should require active public repositories, enough historical issues to estimate repository norms, stable issue-event coverage, and a documented primary language. Pull requests, mirrors, spam, and test repositories are excluded; bot accounts are filtered with declared rules informed by ground-truth bot research [23]. Deleted/unavailable text is recorded as missing rather than inferred. Language identification precedes NLP extraction.

Maintainers are identified at each cutoff using role-bearing repository actions and historically observed merge/triage privileges where available; sensitivity analyses compare definitions. Data collection stores immutable event timestamps, actor pseudonyms, and provenance. Sampling should avoid dominance by a few projects and should report repository-selection attrition.

## B. Escalation Label Construction

Table III separates future indicators from possible leakage. Repository-specific label dictionaries and senior-role mappings are reviewed by knowledgeable annotators. A stratified issue sample is independently coded by multiple annotators using the event timeline after  $t_i^p$ , with disagreements adjudicated and Cohen’s  $\kappa$  (or an appropriate multi-rater coefficient) reported without treating reliability as construct validity. Rules are revised only on development data. Labels should be tested under a narrow high-specificity definition and a broader compound definition.

Table 3: Proposed escalation operationalization.

| Indicator                     | Observable Event/Trajectory                                                 | Labeling Rationale                                               | Leakage Risk                            | Validation Method                                                          |
|-------------------------------|-----------------------------------------------------------------------------|------------------------------------------------------------------|-----------------------------------------|----------------------------------------------------------------------------|
| Reopening                     | Closed issue returns to open state after cutoff                             | Signals failed or incomplete closure, subject to workflow review | Severe if used as predictor             | Manual inspection of reopen reason and state history                       |
| Priority/severity promotion   | Post-cutoff change to repository-validated critical/high label              | Explicit maintainer escalation of handling priority              | Severe; labels are timestamped          | Repository label dictionary plus annotator confirmation                    |
| Senior/core assignment        | Post-cutoff assignment or direct intervention by a validated core actor     | Indicates exceptional expertise or authority may be required     | Role history can leak future status     | Time-indexed role mapping and expert review                                |
| Discussion expansion/conflict | Repository-abnormal growth with sustained disagreement or validated tension | Captures coordination and interpersonal load                     | Later comments cannot be features       | Multi-annotator coding plus trajectory thresholds learned on training data |
| Cross-repository escalation   | Transfer or post-cutoff reference that redirects responsibility             | Observable widening or relocation of coordination scope          | References after cutoff prohibited      | Inspect typed transfer/reference event                                     |
| Abnormal unresolved duration  | Lifetime exceeds comparable repository/issue-type norm                      | Captures prolonged maintenance burden                            | Eventual lifetime is future information | Survival-aware, training-derived norms plus workflow review                |

## C. Observation Windows

Models are trained separately for metadata at creation; the first 6, 12, and 24 hours; the first maintainer response; the first three comments; and the first five eligible interaction events. Clock windows compare equal elapsed time, whereas event windows compare equal information opportunities. Issues that never receive the required event form a documented non-arrival stratum rather than being silently dropped. Sensitivity analysis evaluates the lead-time/information trade-off and whether window choice changes feature importance.

## D. Data Splitting and Leakage Control

Evaluation uses forward temporal train-validation-test splits, preserving order because random within-project validation can misrepresent future performance [29, 30]. A second protocol holds out entire repositories or domains. Repository-grouped stratification prevents one project’s issues from appearing across folds in cross-project tests. Near-duplicate reports are clustered before splitting using text and reference links. All vocabularies, scalars, label thresholds, historical centralities, and contributor histories are fitted from the training past only. No random split is reported as the sole evidence.

## E. Class Imbalance, Metrics, and Alert Utility

Escalation is expected to be a minority operational condition, but its prevalence must be measured rather than assumed. Candidate remedies include class-weighted loss, balanced sampling within training folds, focal loss, and validation-set threshold calibration. Effort-aware quality-assurance research motivates evaluation under a constrained review budget [31]; general imbalance research cautions that resampling and metrics must match the application [32]. Precision-recall analysis is especially informative for imbalanced binary outcomes [33].

For true positives  $TP$ , false positives  $FP$ , and false negatives  $FN$ :

$$Precision = \frac{TP}{TP + FP}, \quad (9)$$

$$Recall = \frac{TP}{TP + FN}, \quad (10)$$

$$F_1 = 2 \frac{Precision \cdot Recall}{Precision + Recall}. \quad (11)$$

Accuracy is insufficient. Reports should include area under the precision–recall curve, ROC–AUC, Matthews correlation coefficient, Brier score, calibration error, recall at fixed alert budgets, and lead time from alert to first qualifying escalation event. Calibration is assessed both overall and by repository/domain.

## F. Baselines, Ablations, and Statistical Analysis

Baselines comprise majority class, metadata-only, text-only, collaboration-only, text-plus-collaboration, and the full model with context. Group ablations remove affect, semantics, graph structure, timing, role history, and workload in turn. RQ2 is answered by paired out-of-time predictions from otherwise matched models, not by comparing unrelated tuning regimes. Bootstrap confidence intervals should resample at the repository level; paired tests compare per-issue or per-repository outcomes as appropriate. Effect sizes accompany adjusted  $p$ -values, and multiple comparisons use a declared correction. Mixed-effects or hierarchical summaries quantify repository variation. No significance or superiority is asserted before these analyses are run.

Table 4: Evaluation design.

| Evaluation Component        | Recommended Design                                                            | Purpose                                            | Threat Addressed                           |
|-----------------------------|-------------------------------------------------------------------------------|----------------------------------------------------|--------------------------------------------|
| Temporal generalization     | Forward train/validation/test periods; all preprocessing fit on training past | Simulate deployment on later issues                | Future-information and concept leakage     |
| Cross-repository robustness | Leave-repository/domain-out evaluation with grouped tuning                    | Measure transfer beyond familiar workflows         | Repository memorization and selection bias |
| Window sensitivity          | Creation, clock, maintainer-response, comment, and event cutoffs              | Quantify lead-time/information trade-off           | Window-dependent conclusions               |
| Duplicate control           | Cluster near-duplicate text and linked issues before splitting                | Prevent template/copied-report leakage             | Inflated generalization estimates          |
| Imbalance and utility       | PR curves, calibrated thresholds, recall at alert budget, lead time           | Align evaluation with limited maintainer attention | Misleading accuracy and alert overload     |
| Ablation                    | Matched metadata, text, collaboration, fusion, and context models             | Isolate incremental predictive information         | Unsupported feature-value claims           |
| Uncertainty                 | Repository-level bootstrap intervals, paired tests, effect sizes, correction  | Quantify estimation and project variation          | Overconfident aggregate conclusions        |

## VI. Discussion

### A. Expected Contributions and Practical Implications

The framework is theoretically justified by two complementary mechanisms. Text approximates the reporter’s problem framing, evidence, and interpersonal stance. Collaboration features capture how the project responds: who joins, how quickly roles coordinate, whether clarification converges, and whether attention becomes concentrated. Consequently, collaboration may add information after controlling for wording and repository workload, but RQ2 deliberately treats that claim as empirical.

Earlier cutoffs maximize lead time but expose little interaction; later cutoffs improve observability while reducing time to intervene. A family of models, rather than a single universal cutoff, lets maintainers match risk tolerance and workflow. Calibrated explanations can improve trust by making a warning contestable. Potential uses include early triage, workload prioritization, at-risk-discussion review, timely moderator or expert involvement, resource allocation, and repository-health monitoring. Predictions support rather than replace judgment; an alert is not evidence of contributor misconduct.

The proposal differs from severity classification because escalation is a future trajectory, and from resolution-time prediction because time is only one normalized outcome indicator. Cross-project evaluation is indispensable: labels, response norms, contributor roles, and discussion cultures vary, so a model that succeeds by memorizing one repository has little operational validity.

## B. Threats to Validity

*Construct validity:* Reopening, priority changes, and long duration can reflect benign workflows. Repository-specific dictionaries, multiple definitions, annotated timelines, and reliability reporting mitigate but do not eliminate this ambiguity.

*Internal validity:* temporal leakage, near duplicates, future contributor status, and post-cutoff normalization can inflate results; immutable cutoffs, train-only transformations, and provenance tests address them. Missing/deleted events, bot errors, and maintainer-role errors require sensitivity analyses and explicit missingness indicators.

*External validity:* active public repositories are not representative of small, private, dormant, or non-English projects. Repository-level sampling, multilingual follow-up, and held-out-project evaluation improve generalizability. Platform practices, labels, automation, and moderation change over time, requiring drift monitoring and periodic revalidation.

*Statistical conclusion validity:* imbalance, threshold tuning, and project clustering can distort aggregate metrics; PR-oriented, calibrated, repository-resampled analysis addresses these risks. Language and cultural bias remains a direct threat because politeness, urgency, and affect vary across communities.

## C. Ethical Considerations

Public availability does not remove research obligations; MSR ethics work emphasizes contextual assessment of data collection and release [34]. The study should minimize data, pseudonymize actors, avoid unnecessary quotation of identifiable comments, secure raw text, and publish only what is needed for reproducibility. Any released dataset should prefer event-derived features or carefully reviewed excerpts over searchable verbatim discussion.

Sentiment and toxicity classifiers may disadvantage non-native English speakers, direct communicators, or marginalized groups. Group auditing should be performed only when ethically and legally supportable, and uncertain affect signals should never trigger automated moderation. Documentation must disclose training domains, error modes, cutoff semantics, and intended use. Maintainers retain oversight; contributors need channels to contest interventions. Automated replies or sanctions based solely on risk scores are outside the intended use because they could chill participation or worsen the discussion the system seeks to protect.

## VII. Conclusion and Future Work

This paper identifies a gap between traditional issue triage, severity, and resolution-time prediction and the early recognition of compound issue escalation. It proposes a leakage-aware framework that fuses early title, description, and comment representations with collaboration dynamics and repository controls. Observable post-cutoff events define escalation, while strict temporal prefixes constrain predictors. The planned evaluation combines multiple observation windows, temporal and cross-repository splits, imbalance-aware and alert-budget metrics, ablations, calibration, and human-validated labels.

If validated, the framework could help maintainers allocate attention and review at-risk discussions without replacing human judgment. Future work should conduct multi-repository and multilingual evaluations; study graph neural networks and dynamic temporal models; separate predictive association from causal effects; and assess human-in-the-loop usefulness through maintainer usability studies. These empirical steps are required before any claim of accuracy, robustness, or operational benefit.

## Author Contributions

I. Wayan Budi Sentana (I.W.B.S): Conceptualization, Methodology, Investigation, Formal analysis, Writing – original draft; Gurdeep Kaur Saminder Singh (G.K.S.S): Conceptualization, Methodology, Validation, Supervision, Writing – review & editing, Project administration.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data Availability

The data that support the findings of this study are available from the corresponding author, G.K.S.S, upon reasonable request.

## Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

During the development of this manuscript, the authors employed AI-assisted tools to support selected aspects of the writing and editing process. AI-assisted technologies were also used selectively during the initial stages of drafting to improve writing efficiency. All AI-generated or AI-assisted material was critically evaluated, revised, and verified by the authors. The authors take full responsibility for the accuracy, originality, interpretation, and integrity of the final manuscript.

## Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## References

- [1] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillère, J. Klein, and Y. L. Traon, “Got issues? who cares about it? a large scale investigation of issue trackers from GitHub,” in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, 2013, pp. 188–197.
- [2] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social coding in GitHub: Transparency and collaboration in an open software repository,” in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, 2012, pp. 1277–1286.
- [3] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “The promises and perils of mining GitHub,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 92–101.
- [4] J. Tsay, L. Dabbish, and J. Herbsleb, “Let’s talk about it: Evaluating contributions through discussion in GitHub,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 144–154.
- [5] J. Anvik, L. Hiew, and G. C. Murphy, “Who should fix this bug?” in *Proceedings of the 28th International Conference on Software Engineering*, 2006, pp. 361–370.
- [6] G. Jeong, S. Kim, and T. Zimmermann, “Improving bug triage with bug tossing graphs,” in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2009, pp. 111–120.
- [7] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, “Predicting the severity of a reported bug,” in *2010 7th IEEE Working Conference on Mining Software Repositories*, 2010, pp. 1–10.
- [8] E. Giger, M. Pinzger, and H. C. Gall, “Predicting the fix time of bugs,” in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.
- [9] R. Kikas, M. Dumas, and D. Pfahl, “Using dynamic and contextual features to predict issue lifetime in GitHub projects,” in *Proceedings of the 13th International Conference on Mining Software Repositories*, 2016, pp. 291–302.
- [10] M. Izadi, S. Akbari, and A. Heydarnoori, “Predicting the objective and priority of issue reports in software repositories,” *Empirical Software Engineering*, vol. 27, no. 2, p. 50, 2022.

- [11] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K. ichi Matsumoto, "Studying re-opened bugs in open source software," *Empirical Software Engineering*, vol. 18, no. 5, pp. 1005–1042, 2013.
- [12] N. Novielli, F. Calefato, and F. Lanubile, "The challenges of sentiment detection in the social programmer ecosystem," in *Proceedings of the 7th International Workshop on Social Software Engineering*, 2015, pp. 33–40.
- [13] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1352–1382, 2018.
- [14] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, and R. Tonelli, "Software development: Do good manners matter?" *PeerJ Computer Science*, vol. 2, p. e73, 2016.
- [15] C. Miller, S. Cohen, D. Klug, B. Vasilescu, and C. Kästner, "Did you miss my comment or what? understanding toxicity in open source discussions," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 710–722.
- [16] I. Ferreira, B. Adams, and J. Cheng, "How heated is it? understanding GitHub locked issues," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 309–320.
- [17] H. S. Qiu, B. Vasilescu, C. Kästner, S. Egelman, C. Jaspan, and E. Murphy-Hill, "Detecting interpersonal conflict in issues and code review: Cross-pollinating open- and closed-source approaches," in *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Society*, 2022, pp. 41–55.
- [18] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2008, pp. 24–35.
- [19] A. Meneely and L. Williams, "Socio-technical developer networks: Should we trust our measurements?" in *Proceedings of the 33rd International Conference on Software Engineering*, 2011, pp. 281–290.
- [20] M. Joblin, W. Mauwerer, S. Apel, J. Siegmund, and D. Riehle, "From developer networks to verified communities: A fine-grained approach," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 563–573.
- [21] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 356–366.
- [22] G. Gousios and D. Spinellis, "GHTorrent: GitHub's data from a firehose," in *2012 9th IEEE Working Conference on Mining Software Repositories*, 2012, pp. 12–21.
- [23] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments," *Journal of Systems and Software*, vol. 175, p. 110911, 2021.
- [24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 4171–4186.
- [25] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019, pp. 3982–3992.
- [26] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "CodeBERT: A pre-trained model for programming and natural languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1536–1547.
- [27] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4765–4774.
- [28] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
- [29] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. ichi Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.
- [30] D. Falessi, J. Huang, L. Narayana, J. F. Thai, and B. Turhan, "On the need of preserving order of data when validating within-project defect classifiers," *Empirical Software Engineering*, vol. 25, no. 6, pp. 4805–4830, 2020.

- [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 Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2013.
- [32] B. Krawczyk, “Learning from imbalanced data: Open challenges and future directions,” *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.
- [33] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” *PLOS ONE*, vol. 10, no. 3, p. e0118432, 2015.
- [34] N. E. Gold and J. Krinke, “Ethics in the mining of software repositories,” *Empirical Software Engineering*, vol. 27, no. 1, p. 17, 2022.