

A Software Engineering Perspective on Documentation Quality: Predicting Newcomer Attraction in Open-Source Projects

Ashraf Abdulmunim Abdulmajeed*

Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq
ashraf_althanoon@uomosul.edu.iq

Abstract Attracting new contributors is fundamental to the long-term sustainability of Open-Source Software (OSS) projects, yet the project-level documentation and community-health characteristics that predict newcomer attraction remain understudied. We present a large-scale empirical study of 717 GitHub repositories, extracting 21 documentation and community-health features (17 documentation/community/issue-communication features plus 4 control variables) and measuring whether each project attracted at least 10 new contributors within six months. Two gradient-boosting models — LightGBM and CatBoost — are evaluated as primary candidates using stratified 5-fold cross-validation, with TabNet included as a deep-learning robustness check rather than a competing candidate, consistent with prior tabular-data benchmarks. CatBoost achieves the highest predictive performance (AUC-ROC = 0.8730 ± 0.0143 , F1-score = 0.8006 ± 0.0179), a statistically consistent though not significant improvement over LightGBM at the five-fold sample size (paired t -test $p = 0.029$; Wilcoxon signed-rank $p = 0.0625$). SHAP analysis shows that issue responsiveness — Issue Close Rate (average SHAP = 0.722) and Average First Response Time (average SHAP = 0.430) — are the strongest predictors of newcomer attraction, with README Length (average SHAP = 0.420) the strongest static documentation predictor; directional analysis further reveals that this association is non-monotonic, peaking for moderate-length READMEs rather than increasing indefinitely with length. An ablation study further confirms that documentation features contribute an incremental AUC gain of +0.0315 above control variables alone and achieve an out-of-fold AUC of 0.7124 in isolation, showing that the documentation signal is not merely a proxy for project size or maturity. A complete replication package is publicly available to support reproducibility.

Keywords Open-Source Software; Newcomer Attraction; Documentation Quality; Machine Learning; Software Repositories; Issue Responsiveness

DOI: 10.19139/soic-2310-5070-4369

1. Introduction

The sustainability of Open-Source Software (OSS) projects depends fundamentally on their capacity to continuously attract new contributors. As of 2024, GitHub hosts over 180 million developers and more than 420 million repositories, making it the largest collaborative software development platform in the world [1]. Despite this unprecedented scale of developer presence, converting interested developers into active project contributors remains a persistent challenge. Projects that fail to attract newcomers face stagnation of development activity, erosion of maintenance capacity, and an elevated risk of abandonment [3, 5, 28]. The question of what project-level characteristics predict successful newcomer attraction is therefore of both practical and scientific significance.

Prior research has examined this challenge from several complementary perspectives. Studies on newcomer onboarding have identified and classified the barriers that prevent potential contributors from making their first contribution, including insufficient documentation, slow issue-response times, and unclear contribution pathways

*Correspondence to: Ashraf Abdulmunim Abdulmajeed (Email: ashraf_althanoon@uomosul.edu.iq). Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq.

[2, 3]. Others have examined the relationship between newcomers' early contribution patterns and their long-term technical success [4], or have investigated how early community participation signals a project's long-term sustained activity [5]. While these contributions have substantially advanced understanding of the onboarding process, they share a common limitation: they analyze newcomer behaviour or project health after the fact, rather than predicting, at the project level, which observable documentation and community-health characteristics are associated with attracting new contributors within a prospective time window.

Documentation quality has been consistently cited as one of the most influential factors in newcomer attraction and onboarding [2, 3]. However, existing studies treat documentation either as a categorical barrier to be overcome or as a static artifact to be classified, rather than as a measurable predictive signal. No prior empirical study has systematically operationalized documentation-quality metrics as features in a machine learning framework to predict whether an OSS project will attract new contributors within a defined and temporally separated observation window. This gap is particularly significant because documentation — unlike social dynamics, maintainer personality, or community culture — is a concrete, measurable artifact that project maintainers can monitor and directly modify, regardless of whether its association with newcomer attraction ultimately proves causal (a question we return to in Section 6.4 and Section 8).

Practically, this gap matters because documentation and issue-tracker responsiveness are among the few project characteristics a maintainer can act on directly: unlike a project's age, star count, or existing contributor base, a README or an issue-triage process can be rewritten, restructured, or sped up within days. A predictive, feature-level account of which documentation and community-health signals precede newcomer attraction — rather than a purely descriptive or retrospective one — therefore has direct, actionable value for maintainers deciding where to invest limited time, in addition to its scientific contribution. To address this gap, we present a large-scale empirical study of 717 GitHub repositories. We adopt a temporal study design with a 12-month feature-measurement window (January 2023 to January 2024) and a six-month prospective target window (January 2024 to July 2024), ensuring strict temporal separation between the features and the outcome variable. We extract 21 features across three documentation and community-health dimensions: README quality, community files, and issue responsiveness. We train and evaluate three machine learning models — LightGBM as a gradient-boosting baseline, CatBoost as the primary model, and TabNet as a deep learning comparator — using stratified 5-fold cross-validation with out-of-fold evaluation to prevent data leakage. The end-to-end pipeline, summarized in Fig. 1, keeps every processing stage — data collection, feature extraction, model training and evaluation, and SHAP analysis — as an independent, reproducible module. We further apply SHAP (SHapley Additive exPlanations) analysis to provide directional, feature-level interpretability, identifying which documentation dimensions are most strongly associated with newcomer attraction and whether their effect is positive or negative. It is important to note that our study is observational in nature; the identified associations between documentation quality and newcomer attraction do not imply causation.

This paper makes the following contributions:

- (1) To the best of our knowledge, we provide the first large-scale empirical study predicting newcomer attraction in OSS projects, with documentation-quality metrics as machine learning features in a dataset of 717 projects from GitHub with a sound temporal design.
- (2) We conduct a comparison of LightGBM and CatBoost as primary gradient-boosting candidates, with TabNet included as a deep-learning robustness check, using stratified 5-fold cross validation and out-of-fold evaluation, ensuring methodological soundness and preventing data leakage.
- (3) We use SHAP to provide explanations and directions to which documentation dimensions are most important for attracting the newcomer, both for dynamic features that are part of the issue responsiveness of the documentation and for static features of documentation.
- (4) We publish a full replication package (data, feature extraction scripts, trained models, and SHAP analysis code) to ensure that the results can be replicated and extended.
- (5) We conduct an ablation study that isolates the contribution of documentation features from control variables, empirically demonstrating that the documentation signal identified by SHAP is not merely a proxy for project size or maturity.

(6) We provide a directional dependence analysis (SHAP dependence and outcome-rate binning) showing that the relationship between README length and newcomer attraction is non-monotonic, peaking at moderate length, and we pair our primary model comparison with a paired significance test to characterize, rather than overstate, the strength of the CatBoost-over-LightGBM performance gap.

The rest of this paper is organized as follows: related work is reviewed in Section 2. Study design and methodology are explained in Section 3. Results for each research question are provided in Section 4. These findings and implications are discussed in Section 5. Threats to validity are discussed in Section 6. Finally, Section 7 presents the conclusions of the paper and Section 8 indicates directions for future work.

2. Related Work

Our work bridges three different research areas: newcomer recruitment and induction in OSS projects, documentation quality and effect in OSS communities and machine learning in software engineering. Each one of them is reviewed below and our work is compared with the works already published in the literature.

2.1. Newcomer Attraction and Onboarding in OSS Projects

New contributor acquisition and recruitment is an on-going field of empirical study for OSS projects. Rehman et al. [1] studied the contribution activities of newcomer candidates on GitHub, and found that the most important obstacle to starting a contribution is to “find a way to start”, while only 30% of newcomer candidates eventually make it to the onboarding process of an OSS project. This work provides a grounding in understanding newcomer behaviour, but concentrates on describing newcomer behaviour rather than predicting which projects they are likely to enter.

While Fronchetti et al. [2] looked at the content of existing CONTRIBUTING files and measured how well existing files met fewer than half of the known barriers to onboarding, they found that there is a gap in documentation. Santos et al. [3] did a systematic literature review of 32 studies in the field of software solutions for supporting newcomer onboarding, where it was concluded that recommendation systems were the most frequently used approach. The correlation between pattern of contributions in the career of newcomers and technical success in their later years was explored by Yue et al. [4] and it was found that early contributions that are steady and consistent is highly correlated with technical success in later years.

Xiao et al. [5] used an XGBoost model based on early-participation metrics for a set of GitHub projects to predict projects that eventually fail, showing that machine learning models can be used to model OSS community dynamics at scale. Assavakamhaenghan et al. [6] took a look at more than 2.7 million initial contributions and determined that the timeliness of initial responses is a significant factor in determining whether new contributions will be made. This empirical evidence directly motivates our inclusion of Avg. First Response Time and Issue Close Rate as predictive features.

Turzo, Sultana and Bosu [7] analyzed the onboarding recommendation systems for new users of OSS, and Gerosa et al. [8] revisited the motivations of contributors in OSS, noting changes in contributors’ motivations as they gain experience. Some researchers investigated current issues faced by contributors to large OSS projects [9] and built a model of sense of virtual community among Linux Kernel contributors [10]. Calefato et al. [11] explored the inactivity of OSS core developers and Padala et al. [12] studied the impact of gender-biased tools on newcomer experiences.

Our study is the first to model the metrics of the documentation quality as a predictive feature in a machine learning framework to attract newcomers at the project level, instead of reviewing what characterises newcomer behaviour [1] or identify onboarding barriers [2, 3] as studied in the above work.

2.2. Exploring the Quality of Documentation for Open-Source Projects

While documentation has long been known to be a key element in maintaining the health of an OSS community, few studies have been done in a quantitative fashion to understand the connection between characteristics of the documentation and the attractiveness of an OSS community to contributors. Venigalla and Chimalakonda

[13] performed an empirical analysis of README files in repositories available on GitHub, and discovered statistically significant associations between README files’ richness and project meta-characteristics, such as community participation and development activity. This study provides correlational evidence of association between README content and project-health indicators, but doesn’t make documentation a predictor in a machine learning context.

Gaughan et al. [14] examined the addition of README and CONTRIBUTING files to FLOSS projects, and they determined that the READMEs were generally created reactively after a contribution event while CONTRIBUTINGs were created proactively in projects with few contributions. Wang, Wang, and Chen [15] investigated the correlation between README quality and the popularity of the projects on GitHub, showing that READMEs with better structure are correlated with having more stars. Popular repositories take more steps to making the code reproducible, and are more detailed in their documentation [16].

We take our own approach to these studies by assigning a structured quality score to each document on three categories: Readme quality, community files, and issue responsiveness; and by using a machine learning framework to control for the potential confounding factors of project popularity, age, and existing community size to assess the predictive power of each.

2.3. Predicting at the machine level in software engineering

In software engineering, prediction tasks have been used heavily with the application of machine learning. Eluri and co [17] used five classification models to determine whether a newcomer will become a long term contributor (LTC) and found that Random Forest worked best. In a large-scale study on LTC prediction, Bao et al. [18] analyzed a large number of projects on GitHub. Ahasanuzzaman et al. [19] extended this direction by incorporating programming-language knowledge units as features, demonstrating improved LTC prediction while employing SHAP for model interpretation. Both studies predict contributor retention rather than newcomer attraction, and rely on contributor behavioural features rather than documentation-quality metrics.

Grinsztajn et al. [20] conducted an extensive benchmark of deep learning and tree-based models across 45 tabular datasets, demonstrating that tree-based models consistently outperform deep learning architectures on medium-sized tabular datasets. Shwartz-Ziv and Armon [21] similarly showed that XGBoost outperforms deep models across tabular datasets. Their empirical findings provide a principled justification for our choice of CatBoost and LightGBM as primary models; we accordingly include TabNet not as a competing candidate but as an empirical robustness check of the Grinsztajn et al. claim on our own dataset (Section 3.5), and contextualize its underperformance in our experiments (Section 5.1.5).

More broadly, machine learning is increasingly applied to software-engineering platforms and tooling beyond contributor prediction — for example, Rafi et al. [30] recently applied a Naive Bayes classifier within an integrated web-security developer platform, illustrating the wider trend of embedding classification models directly into developer-facing tools, a direction our own Implications for Tool Builders (Section 5.2.3) also points toward. Our work contributes to this line of research by targeting newcomer attraction as a project-level prediction outcome, using documentation-quality features that are directly actionable by maintainers, and providing SHAP-based directional explainability — aspects that distinguish it from prior contributor-centric prediction studies. Table 1 summarizes the key studies discussed above, organized across the three thematic areas: newcomer attraction and onboarding, documentation quality in OSS, and machine learning-based prediction in software engineering.

Table 1. Overview of related studies across three thematic areas

Ref.	Thematic area	Methodology	Dataset	Key features	Key finding	Gap vs. our work
Rehman et al. [1]	Newcomer attraction & onboarding	Descriptive / survey	GitHub newcomers	Contribution activity, barriers	~30% onboard; “finding a way to start” is the main barrier	Characterizes behaviour; no attraction prediction
Fronchetti et al. [2]	Newcomer attraction & onboarding	Content analysis	CONTRIBUTING files	File content vs. barriers	Existing files cover < half of known barriers	Analyzes content; not an ML feature

Continued on next page

Table 1 – continued from previous page

Ref.	Thematic area	Methodology	Dataset	Key features	Key finding	Gap vs. our work
Santos et al. [3]	Newcomer attraction & onboarding	Systematic review	32 studies	Onboarding solutions	Recommenders are the most common solution	Retrospective review; no modeling
Yue et al. [4]	Newcomer attraction & onboarding	Longitudinal study	GitHub histories	Contribution patterns	Early consistency predicts long-term success	Targets retention, not attraction
Xiao et al. [5]	Newcomer attraction & onboarding	ML (XGBoost)	GitHub participation data	Early participation signals	Predicts long-term sustainability	No documentation features
Assavakamhaenghan et al. [6]	Newcomer attraction & onboarding	Empirical / statistical	2.7M+ contributions	Response timeliness	Timely responses → continued contribution	Individual-level; not project ML
Turzo et al. [7]	Newcomer attraction & onboarding	Evaluation study	Onboarding systems	Recommender effectiveness	Improves progression to LTC	Evaluates tools, not documentation
Gerosa et al. [8]	Newcomer attraction & onboarding	Survey / empirical	OSS contributors	Motivations over time	Motivations shift with experience	Motivation dynamics, not attraction
Guizani et al. [9]	Newcomer attraction & onboarding	Qualitative / survey	Large OSS organizations	Contribution challenges	Persistent social/technical challenges	Qualitative; no predictive modeling
Trinkenreich et al. [10]	Newcomer attraction & onboarding	Empirical / survey	Linux Kernel	Sense of community	Belonging → sustained activity	Targets belonging, not attraction
Calefato et al. [11]	Newcomer attraction & onboarding	Survival analysis	GitHub core devs	Inactivity patterns	Inactivity → project decline	Core-dev inactivity, not inflow
Padala et al. [12]	Newcomer attraction & onboarding	Mixed methods	Newcomer experiences	Gender-biased tools	Biased tools harm onboarding	Equity focus, not documentation
Venigalla et al. [13]	Documentation quality	Correlation study	GitHub READMEs	README richness	Richness correlates with engagement	Correlational; no ML framework
Gaughan et al. [14]	Documentation quality	Historical analysis	FLOSS projects	File introduction timing	READMEs minimal; CONTRIBUTING reactive	Descriptive, not predictive
Wang et al. [15]	Documentation quality	Correlation study	GitHub projects	README quality, stars	Better READMEs → more stars	Popularity proxy, not attraction
Fan et al. [16]	Documentation quality	Empirical analysis	AI repositories	Documentation depth	More docs ease reproducibility	AI-specific; no attraction ML
Eluri et al. [17]	ML-based prediction in SE	ML (5 classifiers)	GitHub histories	Behavioural features	Random Forest best predicts LTC	Retention, not documentation
Bao et al. [18]	ML-based prediction in SE	ML (large-scale)	GitHub data	Behavioural features	Strong LTC accuracy at scale	Retention of existing contributors
Ahasanuzzaman et al. [19]	ML-based prediction in SE	ML + SHAP	GitHub + PL units	PL knowledge units	PL units improve LTC; SHAP adds interpretability	Retention, not documentation
Grinsztajn et al. [20]	ML-based prediction in SE	Benchmark (45 datasets)	Tabular datasets	DL vs. tree-based	Tree-based models outperform DL	Justifies model choice; not OSS
Shwartz-Ziv et al. [21]	ML-based prediction in SE	Benchmark study	Tabular datasets	XGBoost vs. DL	XGBoost outperforms DL	Supports model choice; not OSS

3. Methodology

The study design, data collection method, feature extraction, construction of target variables, machine learning models, and evaluation method are described. The replication package is a modular pipeline consisting of four stages of python code: (1) data collection, (2) feature extraction, (3) model training/evaluation, and (4) SHAP analysis, as illustrated in Fig. 1. Each stage is executed as a separate unit and communicates with other stages in a structured file format (csv, json) that allows execution and audit of each of the stages independently. Data collection module fetches repository metadata, READMEs, community files and GitHub issue activity from the REST API; feature-extraction module processes raw API responses into the 21 features outlined in Section 3.3 using out-of-fold (OOF) prediction aggregation for the 3 models; and SHAP module uses TreeExplainer for the tree-based models and KernelExplainer for TabNet to generate both mean-absolute-SHAP (MAST) and directional beeswarm visualizations. This pipeline is part of a complete replication package, which is described further in the Data and Software Availability statement that precedes the References.

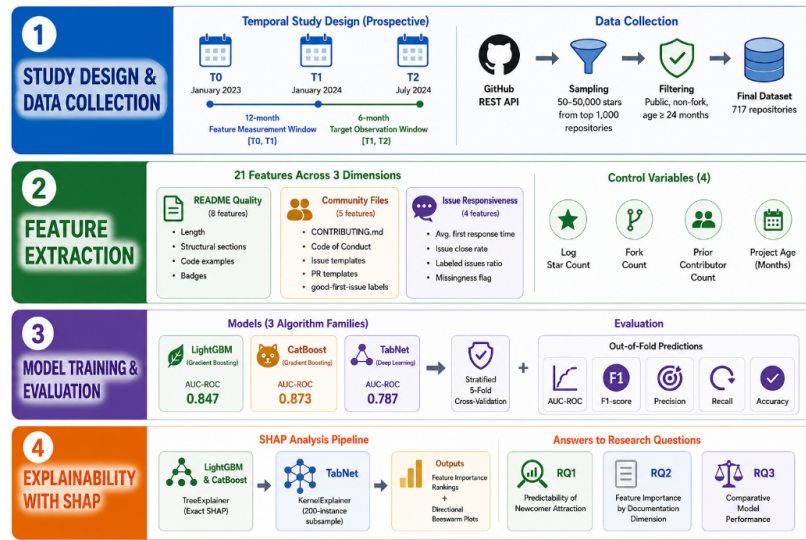


Figure 1. End-to-end methodology pipeline for predicting newcomer attraction in OSS projects.

3.1. Study Design and Temporal Framework

We adopt a prospective temporal study design to ensure strict separation between the feature-measurement window and the target-observation window, thereby preventing temporal data leakage. The design is organized around three reference points: T0 = January 1, 2023 (start of the feature-measurement window); T1 = January 1, 2024 (cutoff point separating features from the target); and T2 = July 1, 2024 (end of the target-observation window). All documentation and community-health features are measured over the 12-month window [T0, T1]. The target variable — the number of new contributors — is counted over the prospective six-month window [T1, T2]. This design ensures that no information from the target window leaks into the feature space.

3.2. Data Collection

We collected repository data from GitHub using the GitHub REST API. Our sampling strategy targeted active, non-trivial, non-forked public repositories that had achieved at least a minimal level of community recognition. Repositories were included if they satisfied all of the following criteria: (1) star count between 50 and 50,000; (2) public repository; (3) not a fork; (4) project age of at least 24 months; and (5) at least one push activity within the 12 months preceding collection.

An initial cap of 1,000 repositories was set, sorted by descending star count. After applying the inclusion and exclusion criteria, 748 repositories remained. Following an inner join with the contributor dataset, 31 repositories were dropped owing to missing contributor data, resulting in a final dataset of 717 repositories. We note that the star-based sampling may introduce a selection bias toward more established projects; this limitation is discussed in Section 6.

3.3. Feature Extraction

We extract 21 features in total — 17 organized into three conceptual documentation and community-health dimensions (README quality, community files, issue communication), plus 4 control variables (Section 3.3.4); this $17+4=21$ breakdown is used consistently throughout the paper, including in the ablation study of Section 4.4 —, all measured over the feature window $[T_0, T_1)$.

3.3.1. README Quality (8 features) README files serve as the primary entry point for potential contributors evaluating whether to engage with a project [13, 14]. We extract: `readme_exists` (binary), `readme_length` (word count), `has_installation_section`, `has_usage_section`, `has_contributing_section`, `has_license_section` (all binary), `has_code_examples` (binary), and `has_badges` (binary). All eight README-quality and five community-file features are point-in-time snapshots taken at T_1 , whereas the four issue-communication features (Section 3.3.3) are aggregated over the full $[T_0, T_1)$ measurement window; this distinction is summarized in Section 3.3.4.

3.3.2. Community Files (5 features) Community-health files signal project governance and contributor-friendliness [2]. We extract: `has_contributing_file`, `has_code_of_conduct`, `has_issue_template`, `has_pr_template`, and `has_good_first_issue` (all binary).

3.3.3. Issue Communication (4 features) Responsiveness of maintainers to issues has been empirically linked to future contributor participation [6]. We extract: `avg_first_response_time` (continuous, hours), `issue_close_rate` (ratio, 0–1), `labeled_issues_ratio` (ratio, 0–1), and `avg_first_response_time_missing` (binary flag). Unlike the README-quality and community-file features, all four issue-communication features are aggregated over issues opened within the full $[T_0, T_1)$ window, rather than measured as a point-in-time snapshot. Of the 717 repositories, 33 (4.6%) had no recorded issue responses in the measurement window; these were imputed using the dataset median (798.65 hours), with a binary missingness flag retained as a separate feature and included in the SHAP analysis (Table 3), so that both the imputed value and the fact of imputation independently inform the model.

3.3.4. Control Variables (4 features) To isolate the predictive contribution of documentation features from confounding project characteristics, we include four control variables: `log_stars` ($\log_{1p}$ of star count at T_1 , controlling for visibility), `log_forks` (controlling for project reach), `project_age_months` (controlling for maturity), and `log_contributors_before_t1` (controlling for existing community size). We apply the $\log_{1p}$ transformation to mitigate right-skewed distributions.

We assessed pairwise multicollinearity among all 21 features and the 4 control variables using Spearman correlation (Fig. 2). The strongest pairwise correlation observed is between Prior Contributors ($\log$) and Project Age (months) ($\rho = 0.54$); among the three popularity-related control variables (`log_stars`, `log_forks`, `log_contributors_before_t1`), the strongest pairwise correlation is $\rho = 0.25$ (`log_stars` $\times$ `log_forks`), and no feature pair in the full matrix reaches the conventional $\rho \geq 0.7$ threshold associated with problematic multicollinearity. We therefore did not apply any additional decorrelation step. Similarly, no capping, winsorization, or transformation was applied to raw feature values such as `readme_length` beyond the $\log_{1p}$ transform used for the control variables and target: CatBoost and LightGBM split on rank-ordered thresholds rather than raw magnitudes, so both models are inherently robust to the heavy right-skew present in several features, and we discuss this design choice further in Section 6.3.

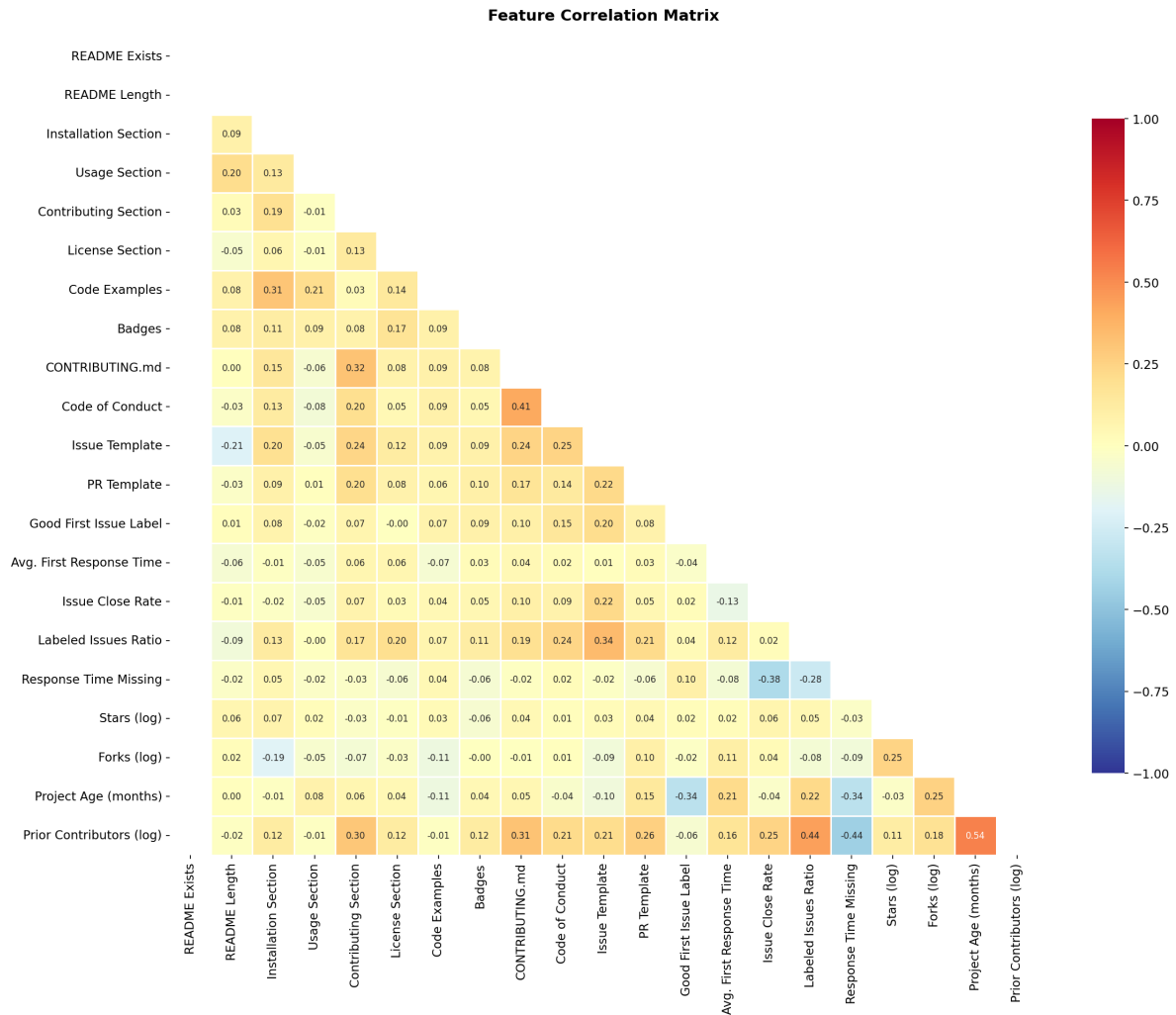


Figure 2. Spearman correlation matrix across all 21 features and 4 control variables. The strongest observed pairwise correlation (Prior Contributors (log) $\times$ Project Age, $\rho = 0.54$) remains well below the $\rho \geq 0.7$ threshold conventionally used to flag problematic multicollinearity.

3.4. Target Variable Construction

The number of first-time contributors that are making at least one commit to the repository in $[T1, T2)$ is the target variable. If a contributor has no commit history in the repository before $T1$ then they are considered a newcomer. The distribution of new contributors in raw terms is very right-skewed (0-468; median = 10). We build a binary classification target by using a threshold that is based on the median: if a project gains 10 or more new contributors, it is considered positive (attracted newcomers = 1); otherwise, it is negative (attracted newcomers = 0). This results in a very well-balanced dataset: 372 positive instances (51.9%) and 345 negative instances (48.1%) as shown in Fig. 3.

3.5. Machine Learning Models

We train and evaluate two primary gradient-boosting models plus one deep-learning robustness check. LightGBM serves as a gradient-boosting baseline, consistent with its use in prior contributor-prediction research [17]. CatBoost

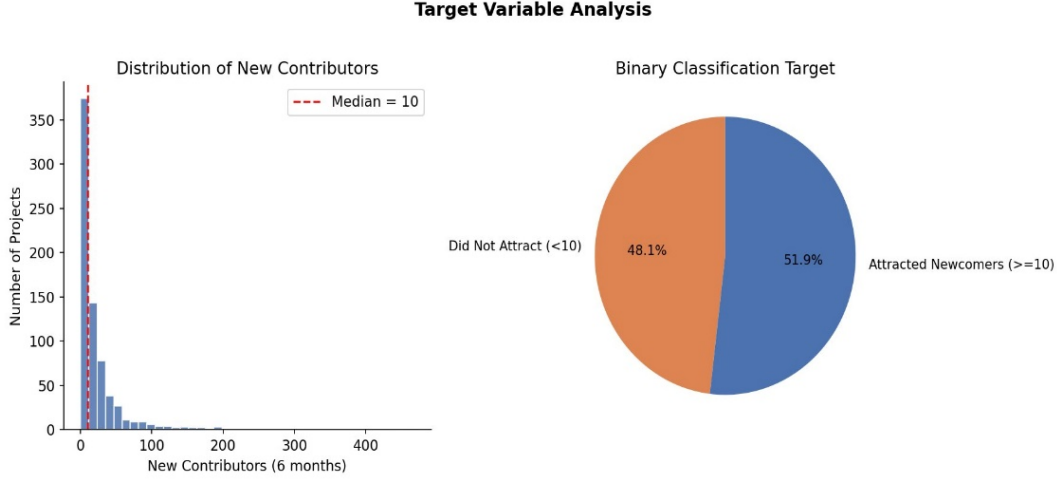


Figure 3. Target variable analysis. Left: distribution of new contributors (6-month window) with median threshold (dashed). Right: binary classification target showing near-perfect class balance (51.9% vs. 48.1%).

is our primary model, selected as a strong gradient-boosting alternative given benchmark evidence that tree-based models generally outperform deep learning on tabular data [20]. TabNet [23] is included not as a third competing candidate but as an empirical robustness check of the tabular-data benchmark claims of Grinsztajn et al. [20] and Schwartz-Ziv and Armon [21] on our own dataset: rather than asking which of three models is best, we use TabNet to ask whether a representative deep-learning architecture can match tree-based performance on this specific problem. Its results are reported alongside LightGBM and CatBoost for transparency (Table 4) but are interpreted separately in Section 5.1.5, including an explicit discussion of the asymmetric hyperparameter tuning between TabNet and the two tree-based models.

3.5.1. Hyperparameter Configuration Table 2 reports the exact hyperparameter configuration used for each model in the primary pipeline (Sections 4.1–4.3). LightGBM and CatBoost use fixed, manually-specified configurations common in prior gradient-boosting studies on comparably-sized tabular datasets, including recent CatBoost applications reporting cross-validated mean $\pm$ standard-deviation performance in a similar style [29]; TabNet uses the default architecture proposed by Arik and Pfister [23] for small-to-medium tabular problems. None of the three models underwent a formal hyperparameter search (e.g., grid or Bayesian search) in the primary pipeline reported in Sections 4.1–4.4; a supplementary Optuna-based search over LightGBM and CatBoost hyperparameters was explored separately and produced comparable out-of-fold AUC (within 0.002), which we report here only as a robustness note rather than as the basis for the main results, since no equivalent search was performed for TabNet. We return to the implication of this asymmetry for the TabNet comparison in Section 5.1.5.

Table 2. Hyperparameter configuration by model (primary pipeline)

Model	Configuration
LightGBM	n_estimators=500, learning_rate=0.05, max_depth=6, num_leaves=31, min_child_samples=20, subsample=0.8, colsample_bytree=0.8, random_state=42
CatBoost	iterations=500, learning_rate=0.05, depth=6, l2_leaf_reg=3, random_seed=42, eval_metric=AUC
TabNet	n_d=16, n_a=16, n_steps=3, gamma=1.3, n_independent=2, n_shared=2, momentum=0.02, mask_type=sparsemax

3.6. Evaluation Strategy

We evaluate all models using stratified 5-fold cross-validation, which preserves the class ratio across each fold. All reported evaluation metrics are computed using out-of-fold (OOF) predictions to prevent data leakage. We report five evaluation metrics: AUC-ROC, F1-score, Precision, Recall, and Accuracy, each as mean $\pm$ standard

deviation across five folds. AUC-ROC is our primary metric, as it measures discriminative ability independently of the classification threshold.

3.7. Explainability via SHAP

We apply SHAP (SHapley Additive exPlanations) analysis [22] to interpret model predictions and identify which documentation dimensions most strongly predict newcomer attraction. For LightGBM and CatBoost, we use TreeExplainer, which computes exact SHAP values in polynomial time [22]. For TabNet, we use KernelExplainer applied to a subsample of 200 instances with 100 background samples. We report both mean absolute SHAP values and directional SHAP analysis from beeswarm plots.

4. Results

The results are organized by the four research questions in this section. All data presented are the means $\pm$ SD of five stratified folds. All of the evaluation plots are created with predictions of out-of-fold (OOF) data.

4.1. RQ1: To What Extent Can Documentation Quality Metrics Predict Newcomer Attraction?

To evaluate RQ1, we compare the predictive performance of the two primary models, alongside the TabNet robustness check, against the majority-class baseline (Accuracy = 51.9%) and the random baseline (AUC = 0.50), with emphasis on AUC-ROC as the primary discriminative metric.

Both primary models exhibit good discriminative power over the random level. CatBoost is the best model with an OOF AUC-ROC of 0.8637 and an AUC of 0.8730 ± 0.0143 with cross validation. LightGBM has an OOF AUC of 0.8485, and the deep-learning robustness check (TabNet) reaches an OOF AUC of 0.7828, discussed separately in Section 5.1.5. Looking at the CatBoost OOF classification report, precision for “Attracted” is 0.81 and recall is 0.79 (F1 = 0.80), for “Did Not Attract” it is 0.78 and 0.80 (F1 = 0.79). The two classes are nearly balanced, indicating that there is no systematic error in the model towards either of the two outcomes. To assess whether CatBoost’s advantage over LightGBM is statistically robust rather than an artifact of fold assignment, we compared the five paired per-fold AUC scores using a Wilcoxon signed-rank test and a paired t -test. CatBoost outperformed LightGBM in all five folds (per-fold AUC: 0.860, 0.880, 0.884, 0.852, 0.889 vs. 0.831, 0.831, 0.862, 0.852, 0.859), which is the strongest pattern of consistent difference obtainable with $n = 5$ paired folds. The Wilcoxon signed-rank test nonetheless does not reach significance at this sample size ($p = 0.0625$, the minimum attainable p -value with $n = 5$ non-tied pairs), while the paired t -test is significant ($p = 0.029$, though its normality assumption is difficult to verify with only five folds). We report both tests rather than a single figure to avoid overstating the certainty either test alone would imply, and return to this limitation in Section 6.1. The ROC curves of all three models with the OOF predictions are shown in Fig. 4.

RQ1 result: Both primary models, and the TabNet robustness check, predict newcomer attraction at substantially above-chance levels, with OOF AUC ranging from 0.78 to 0.86. CatBoost achieves the best discriminative performance (OOF AUC = 0.8637, CV AUC = 0.8730 ± 0.0143).

4.2. RQ2: What are the most significant predictors of newcomer attraction with regards to the Documentation Dimensions?

For RQ2, the two primary tree-based models (LightGBM and CatBoost) are used to compute the ranked mean absolute SHAP values, with TabNet’s values reported separately as a robustness reference (Section 3.5.1, Section 5.1.5). The full SHAP importance results are shown in Table 3. The top-15 features by mean absolute SHAP value are shown in Fig. 6 for all models, and CatBoost beeswarm plot is shown in Fig. 7.

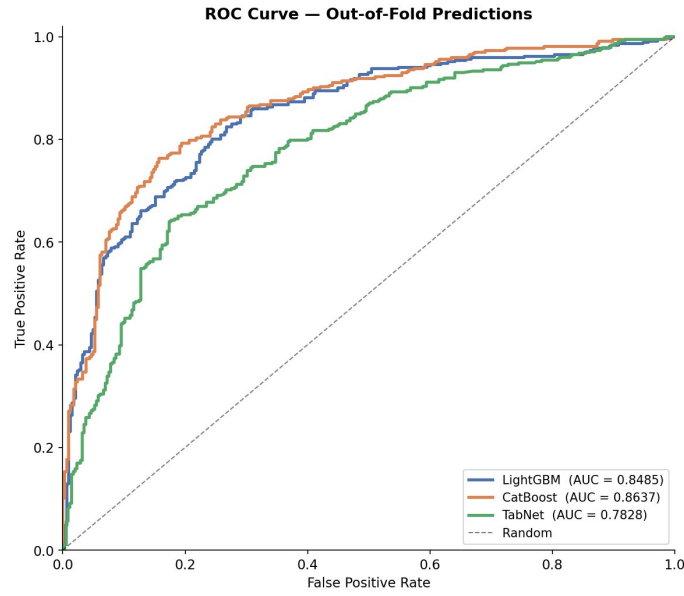


Figure 4. ROC curve — out-of-fold predictions. CatBoost (AUC = 0.8637) consistently outperforms LightGBM (AUC = 0.8485) and TabNet (AUC = 0.7828) across all operating points; the dashed line represents the random-classifier baseline.

Table 3. Mean absolute SHAP values by feature (ranked by LightGBM+CatBoost average; both use exact TreeExplainer). TabNet (approximate KernelExplainer, 200-instance subsample) is reported separately and excluded from the ranking and from the averaging, since its values are not computed on a comparable basis (Section 5.1.5).

Rank	Feature	LightGBM	CatBoost	Avg. (LGB+CB)	TabNet (robustness ref.)
1	Prior Contributors (log)	2.891	2.007	2.449	0.211
2	Project Age (months)	1.924	1.384	1.654	0.148
3	Issue Close Rate	0.861	0.584	0.722	0.057
4	Avg. First Response Time	0.540	0.320	0.430	0.017
5	README Length	0.468	0.372	0.420	0.017
6	Stars (log)	0.487	0.230	0.359	0.022
7	Labeled Issues Ratio	0.380	0.269	0.324	0.021
8	Issue Template	0.324	0.316	0.320	0.033
9	Forks (log)	0.320	0.193	0.257	0.026
10	Usage Section	0.145	0.160	0.153	0.018
11	Installation Section	0.160	0.121	0.140	0.023
12	License Section	0.141	0.099	0.120	0.022
13	CONTRIBUTING.md	0.104	0.108	0.106	0.014
14	Contributing Section	0.134	0.063	0.098	0.019
15	Code of Conduct	0.124	0.042	0.083	0.015
16	Code Examples	0.077	0.036	0.057	0.018
17	Good First Issue Label	0.053	0.052	0.053	0.012
18	PR Template	0.035	0.038	0.036	0.016
19	Response Time Missing (flag)	0.029	0.025	0.027	0.013
20	Badges	0.027	0.025	0.026	0.016
21	README Exists	0.000	0.000	0.000	0.000

Restricting the ranking to the two exact-TreeExplainer models changes only two adjacent pairs relative to a three-model average (Labeled Issues Ratio/Issue Template at ranks 7–8, and Response Time Missing/Badges at ranks 19–20); the top six features and overall ordering are unaffected, indicating the ranking is not sensitive to the inclusion or exclusion of the approximate TabNet explainer.

(Finding 1) Control variables are paramount. The top two features in all models include the control features; Prior Contributors (log) (average SHAP = 2.449) and Project Age (months) (average SHAP = 1.654). These values are far greater than any of the documentation features, and support our control strategy.

(Finding 2) The strongest documentation signal is issue responsiveness. Issue Close Rate (average SHAP = 0.722) ranks third overall and is the strongest documentation predictor. Beeswarm analysis (Fig. 7) shows a clear positive directional pattern: projects with high Issue Close Rate values cluster on the positive side of the SHAP axis, indicating that timely issue resolution is associated with higher newcomer attraction. Avg. First Response Time (average SHAP = 0.430) ranks fourth and shows the opposite directional pattern, with high response-time values clustering on the negative side of the SHAP axis — suggesting that longer response times are associated with lower attraction probability, consistent with Assavakamhaenghan et al. [6].

(Finding 3) README depth is more important than its existence — but the relationship is non-monotonic, not simply “longer is better”. Overall, README Length (average SHAP = 0.420) is the fifth highest-ranking factor by magnitude. By comparison, README Exists (average SHAP = 0.000) has no predictive power for any model, because the variance is almost negligible — almost every repository in our data set either has a README file, or does not.

Unlike Issue Close Rate and Avg. First Response Time, whose beeswarm patterns are cleanly monotonic (Fig. 7), the direction of the README Length signal is not monotonic. A dependence analysis of each project’s SHAP value against its own README Length (Fig. 5a) shows a strong *negative* Spearman correlation between README Length and its own SHAP contribution ($\rho = -0.619, p < 10^{-76}$, CatBoost model): SHAP contributions are positive for short-to-moderate READMEs and increasingly negative for very long ones. Binning projects by README Length (Fig. 5b) confirms this pattern directly in the observed outcome rather than only in SHAP space: the raw attraction rate rises from 37.3% for the shortest READMEs (0–200 words, $n = 51$) to a peak of 62.1% for moderate-length READMEs (500–1,000 words, $n = 214$), then declines steadily to 31.3% for the longest READMEs (8,000+ words, $n = 32$). We therefore revise our interpretation: README Length is the strongest *static* predictor of newcomer attraction by magnitude, but the association is better described as an inverted-U with an apparent optimum around 500–1,000 words, rather than a monotonically positive relationship. We discuss the implications of this revised finding for maintainer recommendations in Sections 5.1.3 and 5.2.1.

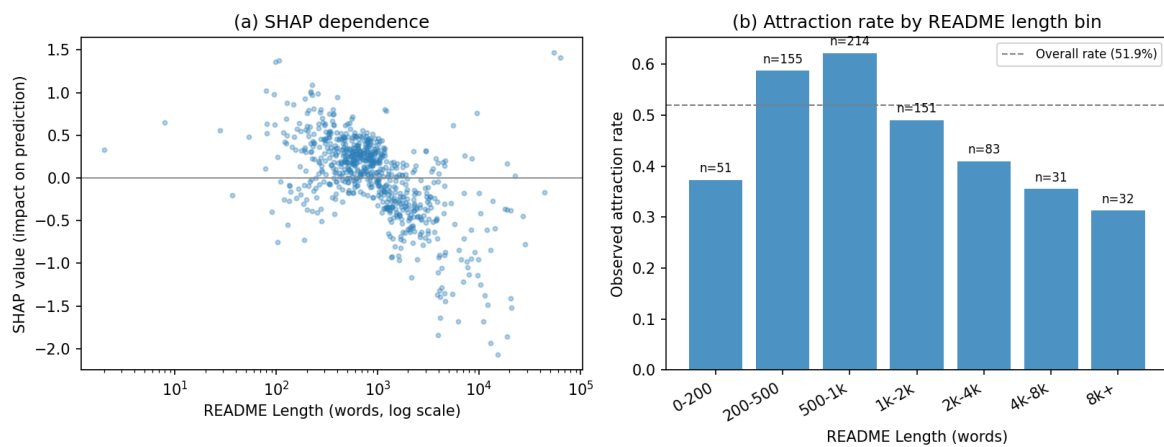


Figure 5. (a) SHAP dependence plot for README Length (CatBoost): each point is one project; the negative slope indicates that the SHAP contribution of README Length turns negative beyond a moderate length. (b) Observed newcomer-attraction rate by README-length bin, confirming a peak at 500–1,000 words followed by a decline for longer READMEs.

(Finding 4) Static community files carry weak signal. CONTRIBUTING.md (avg. SHAP = 0.106), Code of Conduct (avg. SHAP = 0.083), and Good First Issue Label (avg. SHAP = 0.053) all rank in the bottom half of the feature-importance table. Issue Template (avg. SHAP = 0.320) is a notable exception, ranking eighth overall with a positive directional effect.

Finding 5: Rankings of features remain similar across tree-based models. The mutual rank of features for LightGBM and CatBoost is highly correlated (Spearman $\rho = 0.971$, $p < 0.000001$, $n = 21$ features), which helps the convergent evidence of the robustness of the results presented. For reference, TabNet’s ranking correlates more weakly with each tree-based model (LightGBM: $\rho = 0.788$; CatBoost: $\rho = 0.738$), consistent with our treatment of TabNet as a separate robustness check rather than a peer model for ranking purposes.

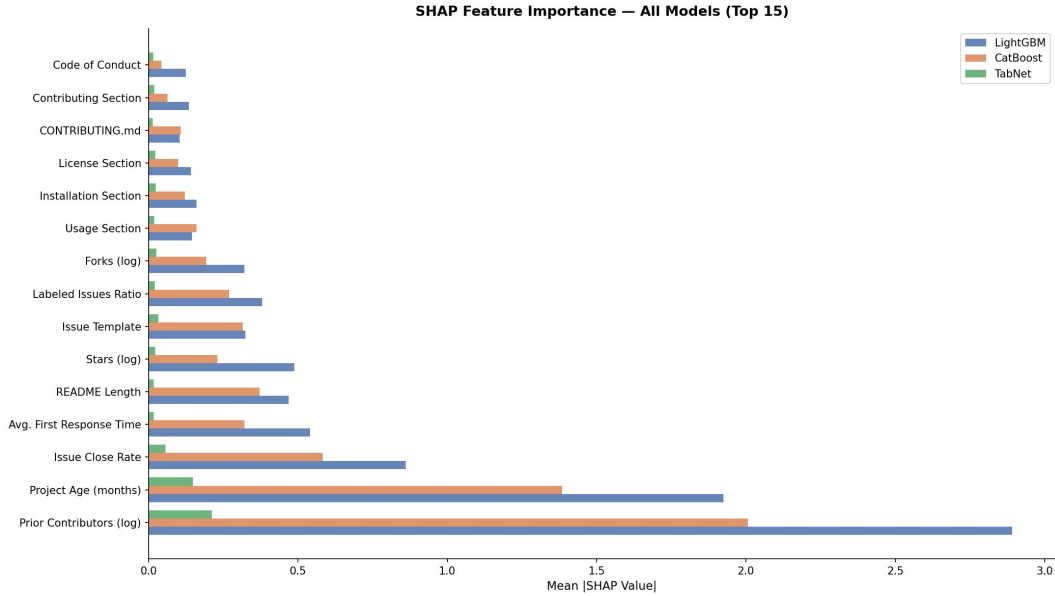


Figure 6. SHAP feature importance — all models (top 15). Mean absolute SHAP values for LightGBM (blue), CatBoost (orange), and TabNet (green).

RQ2 summary: Among documentation dimensions, issue responsiveness — specifically Issue Close Rate (avg. SHAP = 0.722) and Avg. First Response Time (avg. SHAP = 0.430, negative direction) — is the strongest predictor of newcomer attraction. README Length (avg. SHAP = 0.420) is the strongest static documentation predictor by magnitude, though its directional effect is non-monotonic, peaking at moderate length (Fig. 5). Static community files show limited predictive power, suggesting that file presence alone is insufficient without active responsiveness. Whether this documentation signal is independent of project size and maturity, rather than a mere proxy for them, is examined next in RQ4.

4.3. RQ3: How Does CatBoost Compare to LightGBM, and How Does a Deep-Learning Robustness Check Perform?

To answer RQ3, we compare the two primary gradient-boosting candidates, and the TabNet robustness check, across five evaluation metrics using stratified 5-fold cross-validation. Table 4 presents the complete results, Fig. 8 the multi-metric comparison, and Fig. 9 the AUC-ROC distribution across folds. As reported in Section 4.1, CatBoost’s AUC-ROC advantage over LightGBM is directionally consistent across all five folds but does not reach significance under a Wilcoxon signed-rank test at this sample size ($p = 0.0625$), while a paired t -test is significant ($p = 0.029$); we treat CatBoost as the stronger candidate on the balance of evidence while acknowledging this is not a large-sample-certain result.

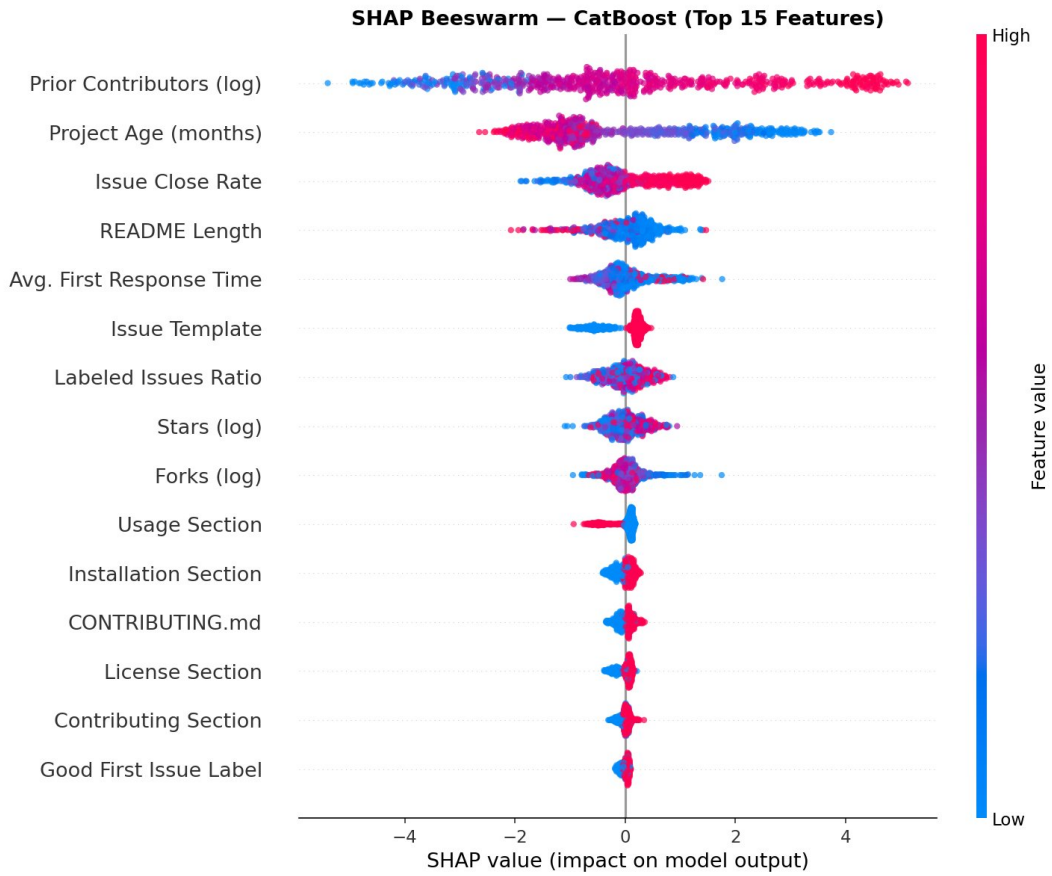


Figure 7. SHAP beeswarm plot — CatBoost (top 15 features). Each point represents one project; colour encodes feature value (red = high, blue = low). Positive SHAP values indicate association with attracting newcomers. Issue Close Rate (high values → positive) and Avg. First Response Time (high values → negative) show the clearest directional patterns.

Table 4. Cross-validation results (mean $\pm$ std, 5 folds)

Model	AUC-ROC	F1-Score	Precision	Recall	Accuracy
CatBoost	0.8730 ± 0.0143	0.8006 ± 0.0179	0.8098 ± 0.0303	0.7930 ± 0.0302	0.7950 ± 0.0192
LightGBM	0.8470 ± 0.0133	0.7803 ± 0.0335	0.7877 ± 0.0356	0.7742 ± 0.0429	0.7741 ± 0.0330
TabNet	0.7871 ± 0.0459	0.7288 ± 0.0501	0.7116 ± 0.0647	0.7525 ± 0.0728	0.7099 ± 0.0526

CatBoost achieves the highest performance across all five metrics. Compared to LightGBM, CatBoost improves AUC-ROC by 2.60 percentage points and F1-score by 2.03 percentage points. Both tree-based models demonstrate stable performance (AUC std: 0.0143 and 0.0133, respectively). TabNet achieves notably lower performance (AUC-ROC = 0.7871, F1 = 0.7288) and exhibits substantially higher variance (AUC std = 0.0459), more than three times that of CatBoost.

RQ3 summary: CatBoost consistently outperforms LightGBM across all five evaluation metrics, achieving the highest AUC-ROC (0.8730 ± 0.0143) and F1-score (0.8006 ± 0.0179), a difference that is directionally robust across all five folds though not significant by Wilcoxon test at $n = 5$ (Section 4.1). LightGBM serves as a strong and stable baseline (AUC = 0.8470 ± 0.0133). The TabNet robustness check exhibits the lowest performance and highest instability (AUC = 0.7871 ± 0.0459), consistent with the known limitations of deep learning on small tabular datasets [20]; Section 5.1.5 discusses this further, including the asymmetric hyperparameter tuning between TabNet and the two tree-based models.

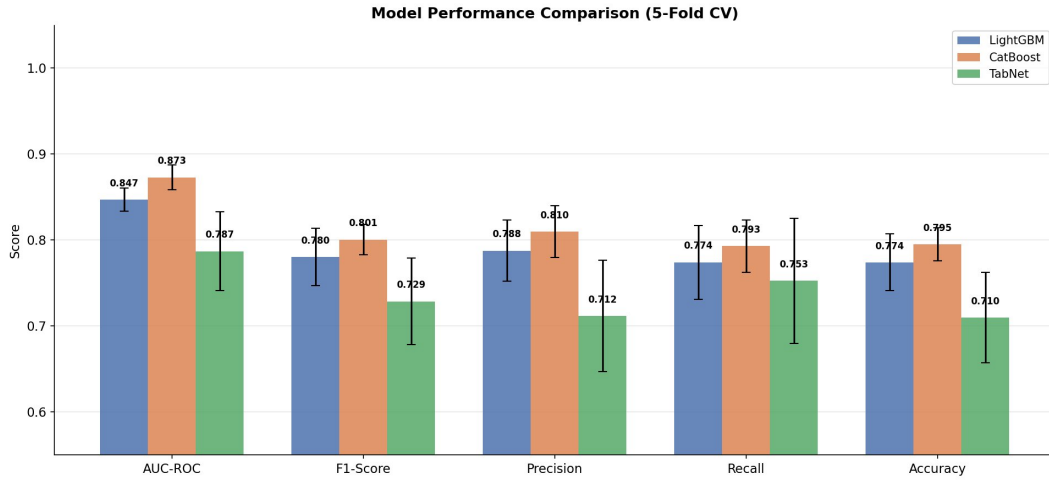


Figure 8. Model performance comparison (5-fold CV). CatBoost (orange) consistently achieves the highest scores across all five evaluation metrics; error bars represent ± 1 standard deviation across folds.

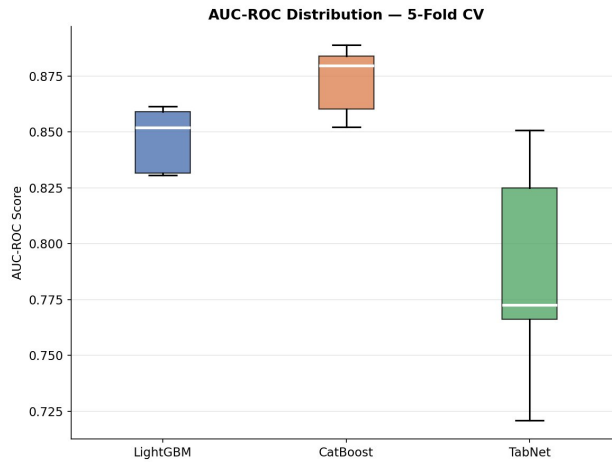


Figure 9. AUC-ROC distribution — 5-fold CV. CatBoost shows the highest median AUC with a compact interquartile range; TabNet's wide box plot indicates high cross-fold instability on this dataset.

4.4. RQ4: Is the Documentation Signal Independent of Project Size and Maturity?

To measure the contribution of features in the documentation, we trained three CatBoost models using different feature sets with the same evaluation conditions (Table 5, Fig. 10). Model A, using only the four control variables, resulted in OOF AUC = 0.8352, which provided a measure of the predictive power of only project size and age. Model B, with all 17 documentation and issue-responsiveness features but none of the controls, had OOF AUC = 0.7124 + 0.2124 points above the random baseline, indicating that there was meaningful standalone predictive signal in the documentation quality features. The full model (Model C) has an AUC of 0.8667, with an incremental AUC of +0.0315 over controls alone. This empirically demonstrates that the documentation features add information over and above that of project size and maturity, directly to the central concern raised regarding a possible overwhelming or dominating of confounding structural characteristics. Also note that this ablation analysis was carried out using a different cross-validation set from the one used in the main experiment reported in Sections 4.1 and 4.3; the slight difference in performance (OOF AUC 0.8667 vs. 0.8637 and CV AUC 0.8777 vs. 0.8730) is due to ordinary run-to-run variation and is not indicative of discrepancies in the underlying model or feature set.

Table 5. Ablation study results — predictive contribution of each feature group (CatBoost, OOF evaluation)

Model	Feature set	N	OOF AUC	CV AUC	OOF F1
Model A	Controls only	4	0.8352	0.8631 ± 0.0206	0.7737
Model B	Documentation only	17	0.7124	0.7352 ± 0.0333	0.7035
Model C	Controls + Documentation	21	0.8667	0.8777 ± 0.0163	0.8021

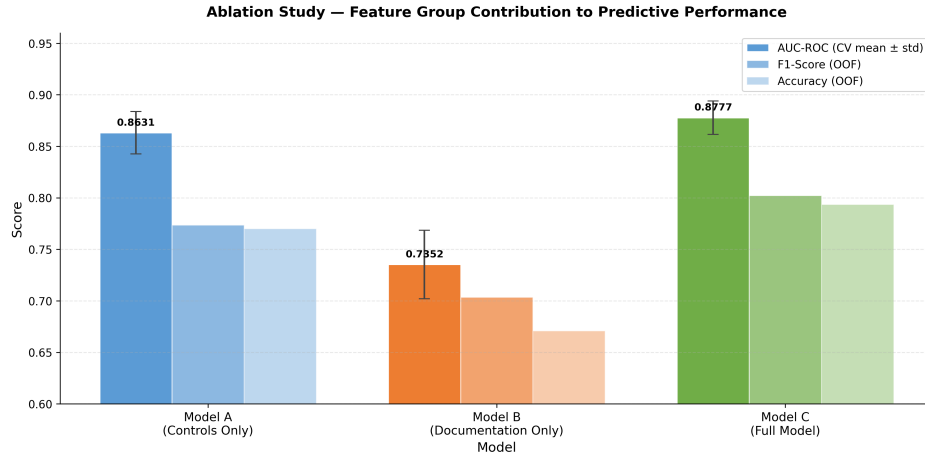


Figure 10. Ablation study — feature group contribution to predictive performance. Each group shows AUC-ROC (CV mean $\pm$ std), F1-Score, and Accuracy under OOF evaluation. Model C (full model) outperforms both ablated variants, with $\Delta\text{AUC} = +0.0315$ over controls alone (Model A) and $\Delta\text{AUC} = +0.1543$ over documentation alone (Model B).

RQ4 summary: An ablation study confirms that documentation features contribute $\Delta\text{AUC} = +0.0315$ above control variables alone and achieve OOF AUC = 0.7124 in isolation, empirically validating that the documentation signal identified in RQ2 is not merely a proxy for project size or maturity.

5. Discussion

This section explains the main findings of the four research questions, their significance for researchers, practitioners and tool builders, and places our results in the context of related work. It should be repeated that our study is observational, with the associations identified not necessarily suggesting causation.

5.1. Interpretation of Key Findings

5.1.1. The impact of the size of established communities The dominance of the two strongest predictors — Prior Contributors (log) (average SHAP = 2.449) and Project Age (average SHAP = 1.654; Table 3) — does not invalidate our documentation findings; rather, it contextualizes them. We agree with the observation, raised during review, that these two control variables are the largest single predictors in the model, and we do not wish to understate this: popularity and maturity are the dominant statistical signal in our data. Our claim is narrower and, we believe, still substantive: that documentation and issue-responsiveness features carry an additional, independent signal on top of that dominant effect, rather than being fully explained by it. Documentation-related features, especially metrics pertaining to the project’s issue-responsiveness, have significant predictive signal even when accounting for the size of the pre-existing community and the age of the project. Thus, Issue Close Rate (average SHAP = 0.722) is third overall (Table 3), implying that responsiveness of the active maintainers is a factor in attracting newcomers beyond project popularity, which is the basis for Stars (log) (average SHAP = 0.359) and Forks (log) (average SHAP = 0.257).

We support this independence claim with three converging pieces of evidence rather than the ablation gain alone. First, the ablation study (Table 5) shows that when documentation features are removed entirely, the model still achieves OOF AUC = 0.8352 on controls alone — but adding documentation yields an incremental gain of $\Delta\text{AUC} = +0.0315$, and documentation features alone (OOF AUC = 0.7124) substantially outperform the random baseline, so the signal does not vanish when popularity and maturity are held fixed. Second, the multicollinearity analysis in Section 3.3.4 (Fig. 2) shows that the strongest correlation between any documentation/community/issue feature and any control variable is $\rho = 0.44$ (Labeled Issues Ratio $\times$ Prior Contributors), still well below the $\rho \geq 0.7$ threshold for problematic multicollinearity, which limits — though does not eliminate — the extent to which documentation could simply be relabeled popularity. Third, the statistically consistent CatBoost-over-LightGBM advantage (Section 4.1) is itself informative here: LightGBM and CatBoost differ in how they exploit interactions and non-linearities among all 21 features together, and CatBoost’s directionally consistent edge across all five folds is at minimum compatible with there being additional, non-trivial structure in the non-control features for the model to exploit. None of these three pieces of evidence, individually, proves that documentation causes attraction — our design remains observational — but together they support treating documentation as a real, independent, if smaller, signal alongside popularity and maturity, rather than as a redundant proxy for them.

5.1.2. The use of issue responsiveness provides many advantages over static documentation The findings that Issue Close Rate and Avg. First Response Time align with previous empirical findings on relationships between issue responsiveness and contributor behaviour [6] are consistent with previous empirical evidence and compare favourably to other competitors (Table 3, Fig. 7). One possible explanation for this is that prospective contributors aren’t evaluating a project statically; a README that is outdated or incorrect can be ignored, while the activity in the issue tracker can be used to gauge maintainer activity and community health. This is consistent with the notion of repository signals referred to in [13] studied by Venigalla and Chimalakonda.

5.1.3. README Depth Versus README Existence: Evidence for a Sweet Spot, Not “Longer Is Always Better” With no predictive power of README Exists, and a meaningful signal of README Length, there is an important nuance: it’s not that there is a README, but that it matters how deep it is. This is in line with Gaughan et al. [14] who have found that initial READMEs in FLOSS projects are light on content.

However, our dependence analysis (Fig. 5, Section 4.2 Finding 3) shows that the relationship is not simply monotonic: the observed attraction rate rises with README length only up to roughly 500–1,000 words, after which it declines steadily, reaching its lowest value for the longest READMEs in our sample (8,000+ words). We interpret this as evidence for a practical sweet spot rather than an open-ended benefit to length. A plausible reading is that a README of moderate length signals a maintained, reasonably documented project without yet imposing a high reading cost on a prospective contributor, whereas very long READMEs may reflect accumulated, poorly curated documentation, or may simply be harder for a newcomer to extract an actionable next step from — consistent with Rehman et al.’s [1] finding that “finding a way to start” is newcomers’ primary barrier. We do not have a content-quality measure that would let us distinguish these explanations, and we return to this as a limitation in Section 6.3 and as a direction for future work using semantic, LLM-based documentation-quality measures. What our data support is the narrower, still actionable claim that projects with substantive but reasonably concise READMEs are more attractive to new participants than projects with either minimal or excessively long ones.

5.1.4. Static Community Files: Presence Is Not Sufficient The relatively weak predictive signal of static community files — CONTRIBUTING.md (avg. SHAP = 0.106), Code of Conduct (avg. SHAP = 0.083), Good First Issue label (avg. SHAP = 0.053; Table 3) — suggests that their mere presence is insufficient as a standalone signal of project attractiveness. Our findings do not contradict the value of these files in facilitating onboarding once newcomers have chosen to contribute [2]. Rather, they suggest that newcomers evaluating a project may weight dynamic responsiveness signals more heavily than static file presence.

5.1.5. TabNet as a Robustness Check: Underperformance and Its Caveats TabNet (AUC = 0.7871 ± 0.0459) underperforms tree-based models by approximately 8–9 percentage points. This is consistent with Grinsztajn et al. [20] and Shwartz-Ziv and Armon [21], who demonstrated that gradient boosting consistently outperforms deep learning on small-to-medium tabular datasets. Our dataset of 717 repositories is substantially below the threshold at which deep-learning advantages typically emerge.

We treat this result as corroborating the Grinsztajn et al. benchmark on our own data rather than as a fair three-way competition, for three reasons worth making explicit. First, hyperparameter tuning was not symmetric across models: LightGBM and CatBoost use fixed, commonly-used configurations (Table 2), and a supplementary Optuna search over these two models’ hyperparameters (Section 3.5.1) changed their out-of-fold AUC by less than 0.002, suggesting they were already close to a local optimum for this dataset; no equivalent search was performed for TabNet, whose architecture (Table 2: $n_d = n_a = 16$, $n_{steps} = 3$) uses the default configuration recommended by Arik and Pfister [23] for small-to-medium tabular problems rather than a configuration tuned specifically to $n = 717$. We cannot rule out that a tuned TabNet architecture would narrow the gap, though prior benchmark evidence [20, 21] suggests tree-based models would likely remain ahead at this sample size regardless. Second, on architecture sizing specifically, $n = 717$ with 21 features is well below the sample sizes at which TabNet and comparable attention-based tabular architectures are typically shown to be competitive [20], so some part of the gap is plausibly attributable to dataset scale rather than to TabNet’s implementation. Third, TabNet’s substantially higher cross-fold variance (AUC std = 0.0459, more than three times that of CatBoost) is consistent with, though not conclusive proof of, incomplete convergence on a dataset of this size; we did not examine per-fold training curves to confirm this directly, and flag it as a limitation of our robustness check rather than a settled conclusion. We report TabNet’s SHAP values in Table 3 for completeness but exclude them from the primary feature ranking (Section 4.2) precisely because of this asymmetric tuning and higher instability, and because its SHAP values are computed via approximate KernelExplainer on a 200-instance subsample rather than the exact TreeExplainer used for LightGBM and CatBoost.

5.2. Implications

5.2.1. Implications for OSS Maintainers Our findings yield several actionable recommendations, stated here qualitatively since our features measure structural properties rather than calibrated thresholds (Section 6.3). First, prioritize issue responsiveness over documentation coverage: maintaining an active issue-triage process — responding promptly and closing resolved issues — is more strongly associated with newcomer attraction than the presence of any static documentation artifact. Second, aim for a substantive but reasonably concise README rather than maximizing length: README Length is the strongest static documentation predictor by magnitude (avg. SHAP = 0.420), but our dependence analysis (Section 5.1.3, Fig. 5) indicates this benefit peaks at moderate length and reverses for very long READMEs, so maintainers should treat “add more content” as good advice only up to a point, favoring clarity and a well-organized structure over exhaustive length. Third, use structured issue templates: among community files, Issue Template shows the strongest predictive signal (avg. SHAP = 0.320). These recommendations are compatible with the tooling developed by Guizani et al. [25] and Qiu et al. [26], whose maintainer dashboards help monitor community-health signals, and could plausibly be integrated into continuous-integration workflows alongside existing dashboard tools [25, 26, 27].

5.2.2. Implications for Researchers Our results have methodological implications for future empirical work. The strong SHAP signal of issue responsiveness relative to static documentation suggests that future feature engineering for newcomer-related prediction tasks should prioritize dynamic interaction features over binary file-presence features. Our temporal study design — with strict feature-target window separation — represents a methodological template that future predictive studies of OSS community dynamics should adopt to prevent temporal leakage. More broadly, this framework opens several research questions that were not previously addressable at this level of operationalization: whether improvements in documentation causally affect newcomer inflow, whether responsiveness norms vary systematically across programming-language ecosystems, and whether documentation-based risk signals are precise enough to support real-time project-health dashboards.

5.2.3. Implications for Tool Builders The 21 features used in our study are all extractable via the GitHub REST API without requiring manual annotation or natural language processing. This suggests that our predictive framework could be operationalized as an automated project-health assessment tool [27]. A scoring system based on the weighted SHAP values — prioritizing issue responsiveness and README depth — could flag repositories at risk of low newcomer attraction, producing a scored report that allows maintainers to take targeted corrective action [3], with the underlying pipeline, trained models, and SHAP analysis code already available in the replication package described at the end of this paper.

5.3. Comparison with Prior Work

Our work differs from prior studies that predict contributor retention [17, 18, 19] by targeting newcomer attraction — the inflow of first-time contributors — rather than the retention of existing contributors. Compared to Xiao et al. [5], who use XGBoost and LIME to predict long-term project sustainability, our work focuses on a more specific and short-term outcome (six-month newcomer count) and employs SHAP rather than LIME, providing directional, game-theoretic feature attributions. The finding that issue responsiveness is the strongest documentation predictor partially aligns with the characterization in Rehman et al. [1] that “finding a way to start” is the primary barrier for newcomers.

6. Threats to Validity

Threats to validity are classified into four types according to Wohlin et al. [24].

6.1. Internal Validity

Threat 1 — Data leakage. All features are used only in [T0, T1), and all evaluation metrics are based on out-of-fold predictions with no information leakage on the within-fold predictions.

Threat 2: Bias due to missing values. 4.6% (33) of the repositories did not have an `avg_first_response_time`. For various reasons, missingness data was imputed using median imputation, and a binary missingness flag was provided to maintain the information about non-random missingness.

Threat 3: Limited power of statistical significance tests at $n = 5$ folds. We applied both a Wilcoxon signed-rank test and a paired t -test to the five per-fold AUC-ROC scores for CatBoost versus LightGBM (Section 4.1). CatBoost outperformed LightGBM in all five folds — the strongest pattern obtainable with five paired observations — yet the Wilcoxon test does not reach significance at this sample size ($p = 0.0625$, the minimum p -value attainable with $n = 5$ non-tied pairs), while the paired t -test is significant ($p = 0.029$) under an unverifiable normality assumption. This is a genuine limitation of the five-fold design rather than a gap we failed to check: with only five paired observations, no non-parametric test can reach conventional significance even under a perfectly consistent effect. Repeated k -fold cross-validation (e.g., 5×5 or 10×10 repeats) would substantially increase the number of paired observations available to such tests and is a natural extension for future work.

6.2. External Validity

Threat 1 — GitHub-only dataset. Our data is limited to GitHub, and may not reflect all environments of OSS development. Projects on GitLab, Bitbucket or self-hosted platforms are not included.

Threat 2 — Selection bias – towards established projects. The repositories selected were done in order of decreasing number of stars, between 50 and 50,000. We may not be able to extrapolate to early stage and low visibility projects. The 50,000-star upper bound was a deliberate design choice, not an arbitrary cutoff: a small number of “mega-projects” (e.g., very high-visibility frameworks) attract corporate-backed, professionally staffed maintenance teams and newcomer pipelines (dedicated onboarding programs, paid contributors, large-scale community-management operations) that plausibly follow different dynamics from the majority of community-maintained OSS projects our study targets; including them risked a small number of high-leverage outliers

dominating the fitted relationships. We acknowledge this means our findings characterize typical, community-maintained OSS projects and should not be extrapolated to either very early-stage repositories or the small set of mega-projects outside our sampling window, and we have kept this qualification in mind in how we phrase our claims throughout the paper.

Threat 3 — Programming-language heterogeneity. There was no language filter and the data set was multi-language. Documentation conventions and community norms can differ significantly from one programming-language ecosystem to another. We did not collect a programming-language field for the 717 repositories in this study and are therefore unable to conduct a post-hoc stratified analysis by ecosystem; pooling across languages was consequently a data-availability constraint as much as a design choice. We consider ecosystem-stratified replication (e.g., comparing systems languages, web-application languages, and data-science languages separately) an important direction for future work (Section 8), and we would expect — though cannot confirm from the present data — that the broad ranking of issue-responsiveness over static file presence is more likely to be ecosystem-invariant than the precise SHAP magnitudes or the exact optimal README length identified in Section 5.1.3.

6.3. Construct Validity

Threat 1 — Operationalisation of “newcomer.” We consider a newcomer to be any contributor who has made at least one commit in this repository after T1 with no commit history in that repository prior to that. This does not include contributors that contribute without committing code (in the form of an issue report or code review). We adopted this narrower, commit-based definition deliberately rather than the broader definition used by some prior work (e.g., Rehman et al. [1], who count any first interaction, including issue reports and comments): a merged or pushed commit is a verifiable, unambiguous signal of code-level contribution that does not require judgment calls about which non-code interactions count as genuine engagement versus passive activity (e.g., a single drive-by comment). The direction of the resulting bias is, we believe, predictable: our definition is conservative and undercounts total newcomer interest, since it excludes prospective contributors who engaged (e.g., opened an issue) but did not go on to commit within the window. It should not inflate attraction counts. We see this as a reasonable trade-off between construct breadth and measurement precision, though we agree with reviewers that it is a narrower construct than “newcomer interest” in the broadest sense, and we flag this explicitly rather than presenting our counts as a complete census of newcomer engagement.

Threat 2 — binary target threshold. Median based threshold (10 new contributors) is an accepted dataset-specific threshold. However, a project with 9 contributors is classified as negative, and one with 11 contributors is classified as positive — this doesn’t necessarily mean that there is a meaningful difference in the real world. We chose the sample median primarily for statistical convenience (it produces a near-balanced binary target, avoiding the additional class-imbalance confound that an arbitrary fixed cutoff could introduce) rather than as a claim that 10 contributors is a universally meaningful definition of “successful” attraction independent of project context: a threshold appropriate for a small utility library is unlikely to be appropriate for a widely used framework. We therefore treat our target variable as a relative, within-sample measure of attraction success rather than an absolute, cross-ecosystem standard, and we would expect the specific numeric threshold — though plausibly not the broader ranking of predictive features — to shift if this design were replicated on a differently-scaled sample. We did not conduct a formal threshold-sensitivity analysis (re-fitting models at, e.g., 8, 9, 11, and 12 contributors) in this revision and flag this as a natural robustness check for future work rather than a claim we can currently support empirically.

Threat 3 — Measure features using the GitHub API. All features are based on the GitHub REST API, which could be rate limited and impact the completeness of the data. Any feature that is compared, like `readme_length`, is a measure of quantity (specifically, word count, not character count, as clarified in Section 3.3.1), not quality of content. Our revised README Length finding (Section 5.1.3) is a direct illustration of why this distinction matters: because we measure only length and not content quality, we cannot distinguish a well-organized 700-word README from a rambling one of the same length, and the inverted-U pattern we observe may partly reflect that very long READMEs are, on average, less well-curated rather than length itself being harmful. Similarly, Issue Close Rate measures closure frequency, not whether issues were closed because they were resolved versus dismissed (e.g., as “wontfix”); a project that closes valid bug reports aggressively without resolving them would

score identically, on this feature, to one that resolves them promptly. We do not have data to distinguish these cases and flag semantic, content-aware documentation- and issue-quality measures (e.g., via LLM-based assessment) as an important direction for future work (Section 8).

6.4. Conclusion Validity

Threat 1 — Uncontrolled selection and mortality of subjects. We come up with associations between documentation-quality metrics and newcomer attraction in our study. Does not make cause/effect statements. In order to make causal claims, a controlled experiment or quasi-experimental designs would be required. Notably, the ablation study (Table 5) provides partial empirical evidence that documentation features carry predictive signal independent of project size and maturity ($\Delta\text{AUC} = +0.0315$), partially mitigating — though not eliminating — concerns about confounding (see the fuller three-part argument in Section 5.1.1, which also draws on the multicollinearity analysis of Section 3.3.4 and the CatBoost-vs-LightGBM comparison of Section 4.1).

Threat 2 — Independence of cross-validation folds. Our 717 repositories may share contributors: a single active developer who contributes to multiple sampled projects would appear across different folds, weakening the assumption that folds are independent samples. The contributor-level identifiers needed to quantify this overlap were not retained in our processed dataset (only project-level aggregate counts were preserved), so we are unable to measure the extent of this overlap directly, and we report it here as an acknowledged limitation rather than a quantified one. We do not expect this to be severe given GitHub’s large and diverse contributor base relative to a 717-project sample, but we cannot rule out a modest positive bias in our cross-validation estimates and flag retaining contributor-level identifiers for overlap analysis as a concrete improvement for future replications.

Threat 3 — TabNet instability. TabNet has a high variance across folds ($\text{AUC std} = 0.0459$), which may lead to concerns about its performance estimates on our data set, compounded by the asymmetric hyperparameter tuning discussed in Section 5.1.5. Be careful when using TabNet SHAP values to make conclusions compared with tree-based models; we exclude them from the primary ranking in Table 3 for this reason.

7. Conclusion

While the projects’ documentation and community-health traits are well known to foster community health, they have not been studied from a predictive perspective to explain why they are successful in attracting new members to their projects. In this paper, we filled this void, conducting a large-scale empirical study of 717 GitHub projects, and using machine learning and explainability approaches based on SHAP to foresee whether or not a project will attract at least 10 new contributors over a six-month period.

Our results show that quality of documentation scores before the observation window have meaningful and consistent predictive signal. CatBoost delivers the best results on all of the metrics ($\text{AUC-ROC} = 0.8730 \pm 0.0143$, $\text{F1-score} = 0.8006 \pm 0.0179$) well exceeding the random baseline, a directionally consistent advantage over LightGBM across all five folds, though not significant by Wilcoxon test at this sample size (Section 4.1). LightGBM offers a stable and robust baseline for gradient-boosting ($\text{AUC-ROC} = 0.8470 \pm 0.0133$). As a deep-learning robustness check rather than a third competing candidate, TabNet achieves the lowest accuracy level ($\text{AUC-ROC} = 0.7871 \pm 0.0459$), which is in line with the existing benchmarks that demonstrate the poor performance of deep-learning architectures in small tabular datasets [20], though we note this comparison was not conducted under symmetric hyperparameter tuning (Section 5.1.5). An ablation study further confirms that documentation features contribute $\Delta\text{AUC} = +0.0315$ above control variables alone and achieve OOF $\text{AUC} = 0.7124$ in isolation, empirically validating their independent predictive signal.

The SHAP analysis provides four practical messages. First, issue-responsiveness metrics — Issue Close Rate (avg. SHAP = 0.722) and Avg. First Response Time (avg. SHAP = 0.430, negative direction) — are the strongest predictors of newcomer attraction overall. Second, README depth (README Length, avg. SHAP = 0.420) is the strongest static documentation predictor by magnitude, while README Exists has a near-universal presence in our sample and does not contribute meaningfully to the signal. Third, this README-length effect is non-monotonic: the observed attraction rate peaks for moderate-length READMEs (500–1,000 words) and declines for both very

short and very long ones, so “longer” is not itself the actionable message — “substantive but concise” is. Fourth, the presence of certain static community files (like CONTRIBUTING.md and Code of Conduct) have only shown limited predictive power, indicating that it is not enough to simply be there.

Collectively, these results indicate that the ability to attract newcomers is more strongly associated with the responsiveness of the maintainers than with the static documentation artifacts a project produces, although both carry independent signal (Section 5.1.1). OSS maintainers seeking to attract more new contributors are encouraged to go beyond the adoption of suggested community-file templates, and to actively manage issues, respond promptly, and keep the README substantive but focused rather than exhaustive. To reiterate, the associations we have found in our study are only observational, and do not necessarily indicate causation. To our knowledge, this is the first empirical study to propose a prediction model for newcomer attraction in OSS projects based on documentation-quality metrics as machine learning features in a directionally explainable fashion using SHAP.

8. Future Work

Causal analysis. The most immediate limitation of our observational design is the inability to establish causal relationships. Future work could apply quasi-experimental methods — such as difference-in-differences analysis applied to projects before and after adopting specific documentation practices — to move toward causal claims.

Cross-platform replication. Our dataset is limited to GitHub repositories. Future studies should replicate our findings on GitLab, Bitbucket, and Apache Software Foundation projects to assess generalizability.

Ecosystem-level analysis. Future work could examine ecosystem-level effects — whether the documentation norms of a dominant package influence newcomer-attraction patterns across dependent packages.

Natural-language quality of documentation. Our features measure structural properties of documentation rather than semantic quality. Future work could incorporate large-language-model-based quality assessments to capture whether documentation content is genuinely informative.

Longitudinal dynamics. Our study uses a single 12-month feature window. Future work could examine how documentation quality evolves over a project’s lifetime and whether improvements are followed by increases in newcomer attraction in subsequent windows.

Newcomer retention after attraction. Our outcome variable captures attraction but not subsequent retention. Future work could extend the pipeline to model the full newcomer journey: attraction, first contribution, and continued engagement.

Threshold sensitivity. We did not re-fit our models at alternative newcomer-count thresholds (e.g., 8, 9, 11, 12) in this revision (Section 6.3, Threat 2). A systematic sensitivity analysis — refitting Table 4 across a small grid of thresholds and checking whether the SHAP ranking in Table 3 is stable — would directly strengthen the construct validity of our binary target and is a natural, low-cost extension of the present replication package.

Contributor overlap across projects. Our cross-validation folds assume independence across the 717 sampled repositories, but we could not verify the extent of contributor overlap between projects with the project-level aggregate data retained in this study (Section 6.4, Threat 2). Future work should retain contributor-level identifiers to quantify this overlap and, if necessary, apply group-aware cross-validation splitting.

Data and Software Availability

The raw dataset, processed dataset, feature extraction pipeline, trained models (LightGBM, CatBoost, TabNet), and SHAP analysis code are all shared in a complete replication package publicly available from <https://github.com/Ashraf-Althanoon85/Impact-on-Newcomer-Onboarding-in-Open-Source-Projects>, licensed as MIT. All global configuration parameters are stored in one file for exact replication; these include the temporal boundaries (T0, T1, T2), repository-sampling thresholds, the newcomer threshold, the random seed and the cross-validation settings. The processed dataset is also available in the repository, and results can be reproduced without having to access GitHub API. The pipeline is created using Python 3.10, NumPy, Pandas, Scikit-learn, LightGBM, CatBoost,

TabNet, SHAP and Matplotlib, and can be reproduced by executing the scripts sequentially. Any questions regarding the replication package should be sent to ashraf_althanoon@uomosul.edu.iq.

Acknowledgements

The author appreciates the facilities and support offered by the University of Mosul, College of Computer Science and Mathematics, through which this research was completed.

Conflicts of Interest

The author declares no conflict of interest.

REFERENCES

1. Rehman, I., D. Wang, R. G. Kula, T. Ishio, K. Matsumoto. *Newcomer OSS-Candidates: Characterizing Contributions of Novice Developers to GitHub*. – Empirical Software Engineering, Vol. 27, 2022, 109.
2. Fronchetti, F., I. Wiese, G. Pinto, I. Steinmacher, C. Treude, M. A. Gerosa. *Do CONTRIBUTING Files Provide Information about OSS Newcomers' Onboarding Barriers?* – In: Proc. of ESEC/FSE 2023, 2023, 16-28.
3. Santos, F., I. Wiese, B. Trinkenreich, I. Steinmacher, A. Sarma, M. A. Gerosa. *Software Solutions for Newcomers' Onboarding in OSS Projects: A Systematic Literature Review*. – Information and Software Technology, Vol. 177, 2025, 107568.
4. Yue, Y., Y. Wang, D. Redmiles. *Off to a Good Start: Dynamic Contribution Patterns and Technical Success in an OSS Newcomer's Early Career*. – IEEE Transactions on Software Engineering, Vol. 49, 2023, No 2, 529-548.
5. Xiao, W., H. He, W. Xu, Y. Zhang, M. Zhou. *How Early Participation Determines Long-Term Sustained Activity in GitHub Projects?* – In: Proc. of ESEC/FSE 2023, 2023.
6. Assavakamhaenghan, N. et al. *Does the First Response Matter for Future Contributions?* – Empirical Software Engineering, Vol. 28, 2023, 75.
7. Turzo, A. K., S. Sultana, A. Bosu. *From First Patch to Long-Term Contributor: Evaluating Onboarding Recommendations for OSS Newcomers*. – IEEE Transactions on Software Engineering, 2025.
8. Gerosa, M., I. Wiese, B. Trinkenreich, G. Link, G. Robles, C. Treude, I. Steinmacher, A. Sarma. *The Shifting Sands of Motivation: Revisiting What Drives Contributors in Open Source*. – In: Proc. of ICSE 2021, 2021, 1046-1058.
9. Guizani, M., A. Chatterjee, B. Trinkenreich et al. *The Long Road Ahead: Ongoing Challenges in Contributing to Large OSS Organizations and What to Do*. – Proc. ACM on Human-Computer Interaction, Vol. 5, 2021, No CSCW2, 407.
10. Trinkenreich, B., K.-J. Stol, A. Sarma, D. M. German, M. A. Gerosa, I. Steinmacher. *Do I Belong? Modeling Sense of Virtual Community Among Linux Kernel Contributors*. – In: Proc. of ICSE 2023, 2023, 319-331.
11. Calefato, F., M. A. Gerosa, G. Iaffaldano, F. Lanubile, I. Steinmacher. *Will You Come Back to Contribute? Investigating the Inactivity of OSS Core Developers in GitHub*. – Empirical Software Engineering, Vol. 27, 2022, No 3, 76.
12. Padala, H. S., C. Mendez, F. Fronchetti, I. Steinmacher et al. *How Gender-Biased Tools Shape Newcomer Experiences in OSS Projects*. – IEEE Transactions on Software Engineering, Vol. 48, 2022, No 1, 241-259.
13. Venigalla, A. S. M., S. Chimalakonda. *Is There a Correlation Between Readme Content and Project Meta-Characteristics?* – Software: Practice and Experience, Vol. 55, 2025, No 3, 589-609.
14. Gaughan, M., M. Champion, J. Hwang, M. Shaw. *The Introduction of README and CONTRIBUTING Files in Open Source Software Development*. – In: Proc. of CHASE 2025, 2025.
15. Wang, T., S. Wang, T.-H. P. Chen. *Study the Correlation between the Readme File of GitHub Projects and Their Popularity*. – Journal of Systems and Software, Vol. 205, 2023, 111806.
16. Fan, Y., X. Xia, D. Lo, A. E. Hassan, S. Li. *What Makes a Popular Academic AI Repository?* – Empirical Software Engineering, Vol. 26, 2021, No 1, 2.
17. Eluri, V., T. Mazzuchi, S. Sarkani. *Predicting Long-Time Contributors for GitHub Projects Using Machine Learning*. – Information and Software Technology, Vol. 135, 2021, 106616.
18. Bao, L., X. Xia, D. Lo, G. C. Murphy. *A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects*. – IEEE Transactions on Software Engineering, Vol. 47, 2021, No 6, 1277-1298.
19. Ahasanuzzaman, M., G. Oliva, A. E. Hassan. *Predicting Long Time Contributors with Knowledge Units of Programming Languages*. – Empirical Software Engineering, Vol. 30, 2025, 99.
20. Grinsztajn, L., E. Oyallon, G. Varoquaux. *Why Do Tree-Based Models Still Outperform Deep Learning on Tabular Data?* – In: Advances in Neural Information Processing Systems (NeurIPS), Vol. 37, 2022, 507-520.
21. Shwartz-Ziv, R., A. Armon. *Tabular Data: Deep Learning Is Not All You Need*. – Information Fusion, Vol. 81, 2022, 84-90.
22. Lundberg, S. M. et al. *From Local Explanations to Global Understanding with Explainable AI for Trees*. – Nature Machine Intelligence, Vol. 2, 2020, No 1, 56-67.
23. Arik, S. O., T. Pfister. *TabNet: Attentive Interpretable Tabular Learning*. – In: Proc. of AAAI 2021, Vol. 35, 2021, No 8, 6679-6687.

24. Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén. *Experimentation in Software Engineering*. Springer, Berlin, 2024.
25. Guizani, M., T. Zimmermann, A. Sarma, D. Ford. *Attracting and Retaining OSS Contributors with a Maintainer Dashboard*. – In: Proc. of ICSE-SEIS 2022, 2022, 36-40.
26. Qiu, H. S., A. Lieb, J. Chou et al. *Climate Coach: A Dashboard for Open-Source Maintainers to Overview Community Dynamics*. – In: Proc. of CHI 2023, 2023.
27. Feng, Z., K. Kimura, B. Trinkenreich, I. Steinmacher, M. Gerosa, A. Sarma. *Addressing OSS Community Managers' Challenges in Contributor Retention*. – ACM Transactions on Software Engineering and Methodology, 2025.
28. Kaur, R., K. K. Chahal. *Exploring Factors Affecting Developer Abandonment of Open Source Software Projects*. – Journal of Software: Evolution and Process, Vol. 34, 2022, No 9, e2484.
29. Ed-Daoudi, R., S. El Khamlichi, B. Ettaki. *A Hybrid Machine Learning Approach for Air Quality Prediction in Morocco: Combining CatBoost with Metaheuristic Optimization Algorithms*. – Statistics, Optimization and Information Computing, Vol. 14, 2025, No 11, 2445-2471.
30. Rafi, M. T. R., M. S. H. Tonmoy, W. Rahman, M. S. Hossain. *WebGuard: Enhancing Web Security Through an Integrated Developer Platform*. – Statistics, Optimization and Information Computing, Vol. 14, 2025, No 11, 2688-2703.