The Popularity Paradigm: How Software Development Metrics Influence Project Fame

Shaik Haseeb Ur Rahman (ID: 924142853), Aayusha Hadke (ID: 924110050),
Celine John Philip (ID: 924165636), Aditi Agrawal (ID: 924167689), Krishna Karthik (ID: 924160880)

1 Introduction & Motivation

In this rapidly evolving field of software engineering, open-source software (OSS) projects have become one of the central forces driving innovation, enabling global collaboration and cost-effective software solutions across industries and academia. As the number of open-source repositories continues to grow, the challenges surrounding a greater number of contributions, adoption, and developers' engagement also intensify [1]. As this is the case, it becomes essential for OSS projects to gain popularity to attract contributors, ensure long-term viability, and maximize their impact [17].

Popularity plays a significant role in the growth and success of open-source projects. It tends to attract a larger pool of developers, which not only enhances the development pace but also brings a diverse technical perspective [3]. This increased visibility helps projects fasten their development process and also gain access to project-specific coaching [2]. Furthermore, the popular OSS repositories contribute significantly to the expansion of software products, both vertically by increasing volume and horizontally by targeting new applications and domains [7] which eventually serve as a valuable resource for researchers and learners [10]. Additionally, developers can also gauge an idea of if their projects are getting accepted and drawing new users by assessing their popularity [6]. Hence, there's a need to analyze the factors that influence the popularity of open-source projects.

Our research aims to investigate how socio-technical metrics and repository quality features influence the popularity of OSS projects. By integrating longitudinal socio-technical data from ASFI repositories with computed normalized popularity scores (pScore), we conduct a comprehensive analysis of behavioral and structural attributes that drive OSS popularity. Unlike existing literature, our work explicitly connects socio-technical constructs with project popularity and also considers repository-level health indicators, offering a more comprehensive study of what makes an OSS project successful.

2 Background

Previous studies did emphasize the importance of evaluating software-related metrics to understand OSS popularity, e.g., the works of [6] concentrated on metrics like number of releases, programming language, and application domain-like metrics, and the work of [11] concentrated on the social interference. However, many of these studies tend to focus heavily on technical aspects or briefly overview the social aspects of the projects as a whole, overlooking the deeper technical and social dynamics that contribute to project success as a whole. Our study bridges this gap and explores how these socio-technical metrics correlate with the OSS Project's popularity. For our study, we have adopted the Apache Software Foundation Incubator (ASFI) projects, which have an extensive community of developers who contribute towards development through commits and e-mails. ASFI provides us with a diverse ecosystem of open-source projects that go through a well-defined incubation and graduation process, making it an ideal environment for us to assess popularity.

2.1 ASFI & Socio-technical Theory

The Apache Software Foundation (ASF) plays a pivotal role in OSS project assessment by fostering structured governance, open collaboration, and community-driven development. OSS projects operate as socio-technical systems (STS), where both social structures (e.g., contributor interactions) and technical aspects (e.g., code contributions) influence popularity and success [2]. Yin et al. (2021) work applies STS modeling to ASF projects, showing that early engagement and communication patterns significantly impact project sustainability [1]. Similarly, socio-technical network models have been used to analyze software development ecosystems in Yin et al. (2021) work, demonstrating that project longevity depends on effective communication and code quality [1]. Our study extends these insights by focusing on popularity, taking metrics such as stars, forks, and pull requests, and assessing how collaboration, engagement, development patterns, and repository quality contribute to project popularity.

2.2 Popularity Score

The metric adopted in this study to assess the project's popularity is referred to as *pScore*, which is defined by Aggarwal, K(2014) [8] as $pScore = stars + forks + pull\_requests^2$. Here *pScore* is the combination of GitHub-based activity indicators, namely the number of stars, forks, and pull requests.

2.3 Contribution of This Study

Our approach is novel in two prominent ways. First, we incorporate both social and technical collaboration indicators to capture developers' behavior more holistically. Second, we extract repository quality dimensions, sourced from the Apache Clutch website, as additional explanatory variables. This combined framework of social, technical, and repository quality of projects provides an extensive understanding of OSS popularity that surpasses prior studies that focus solely on technical metrics or social metrics.

2.4 Research Questions

Our research is guided by three key questions:

- **RQ1:** How do network structures, collaboration dynamics, and project activities evolve over time, and how do these elements influence the popularity of a project?
- **RQ2:** To what extent do developer interaction, collaboration frequency, consistency, and engagement consistency influence the popularity of open-source projects?
- **RQ3:** How is the overall quality of a repository correlated with a project's popularity and success?

3 Methodology

3.1 Dataset

For RQ#1 and RQ#2, we have utilized a comprehensive ASFI dataset from Anonymous, A. [4], which provides a longitudinal perspective on developer coding and communication activities across 330 projects from the Apache Software Foundation Incubator (ASFI). Each ASFI project culminates in one of three outcomes: graduation, retirement, or continued incubation. The dataset includes data from 224 projects that successfully graduated, 69 projects that were retired, and 37 projects that were still in incubation at the time of data collection. This dataset is the foundation of our analysis on the influence of socio-technical metrics on project popularity. For RQ#3, additional data was parsed from the Apache Software Foundation's Incubator Clutch Status website [5]. This dataset contains detailed information about the current podlings under incubation, including their project status, progress within the incubation lifecycle, available resources, and general health indicators [16].

3.2 Features/Metrics of Interest

For our analysis of #RQ1 and #RQ2, we primarily used the features from the recent work of Yin et al. in [1], [3] and Anonymous, A. [4]. Their research examined episodic changes in socio-technical aspects of ASF Incubator Projects, considering socio-technical features, institutional agents, and resistance features. Given our focus on temporal traces of socio-technical features, we incorporated 9 such features from their study. The social features adopted from the aforementioned work include **s_graph_density**, which represents the sparsity of connections among contributors within a project; **s_largest_component**, indicating the largest connected component; **s_weighted_mean_degree**, which measures the weighted mean degree of the social networks; **s_net_overlap**, representing the number of developers consistently active in social networks; **complexity_index**, the product of s_num_component and s_avg_clustering_coef; and **s_num_nodes**, denoting the number of unique active developers in the social networks. The technical features include **t_graph_density**, which, unlike social graph density, reflects the interconnectivity between project files; **t_num_file_per_dev**, measuring the degree of multitasking by files per developer node; and **t_net_overlap**, which captures the number of consistent developers in the technical network. Additionally, in #RQ3 after data extraction features like project reporting frequency, issue tracker presence, mailing lists, distribution readiness, and release infrastructure indicators were used [15].

3.3 Data Preprocessing

- **RQ#1 and RQ#2:** The data of ASFI feature metrics mentioned in section 3.2, were acquired from the work of Anonymous, A. [4]. The dataset retrieved was clean and structured, hence no explicit pre-processing stage was required for this. To calculate the Popularity Score (*pScore*) of all the repositories, we first scraped the number of stars, forks, and pull requests from all ASFI repositories using GitHub token-based authentication and REST API trigger methodology. This was inspired by the work of Weber, S (2014) [12]. During this process, some repositories could not be scraped due to inconsistent API patterns; these were handled manually, and the script was re-executed to successfully compute the *pScore* for all repositories. The calculated *pScores'* were then normalized, and a quantile threshold of 0.3 was applied. This allowed us to classify the top 70% of repositories as popular. Initially, a fixed threshold value (i.e., 0.5) was considered to distinguish between popular and non-popular repositories. However, this approach resulted in a highly imbalanced classification due to the skewed distribution of normalized *pScore* values as most repositories had significantly lower scores, with only a few scoring high. Hence, quantile-based classification was adopted to ensure a balanced dataset. After this process, a comprehensive popularity dataset was generated. To analyze the influence of socio-technical metrics, this dataset was then merged with the ASFI dataset [4], resulting in the final dataset used for RQ#1 and RQ#2 analysis [20].
- **RQ#3:** For RQ#3, web scraping techniques were employed to extract relevant data from the Apache Incubator Clutch Status website [5]. We generated a script that sends HTTP requests and parses the responses using the BeautifulSoup library. Through this, we gathered key software project metrics, which were used to judge the repository quality, such as mailing list availability, issue tracker status, and reporting frequency. Finally, the same methodology was subsequently followed to calculate the *pScore* and define popularity for RQ#3 repositories, generating the final dataset and completing the preprocessing pipeline [20].

3.4 Data analysis

- **Grouped Mean Comparison [RQ#3]**: It is a technique for calculating the mean popularity score of repositories from the occurrence or non-occurrence of various project quality indicators such as release distribution areas, issue trackers, and mailing lists. Comparing and grouping these means will enable us to find potential associations between repository attributes and popularity. Grouping makes pattern recognition easier in the interaction of binary features and popularity.

- **T-Test Analysis with p-Values [RQ#3]**: Independent t-tests ascertain whether differences in the mean pScores of repositories with and without particular quality indicators are statistically significant. The resulting p-values are the probability that observed differences were the result of chance, enabling us to identify which attributes have a significant influence on project success. Several groups might be compared by ANOVA, but the majority of the features (e.g., has_issue_tracker) were binary, so T-tests were more appropriate.

- **Mann-Whitney test [RQ#1 and RQ#2]**: We employed the Mann-Whitney test to contrast network structures, collaboration, and project activity of popular (1) and unpopular (0) open-source projects since it is a non-parametric test which can be used for two-group comparison without assuming normality. Kruskal-Wallis test was also in use, but not necessary since it is used in the case of three or more groups. While the Mann-Whitney U test is effective in revealing differences in distribution, it does not create causality or measure an effect size. Later research can apply regression analysis or time-series analysis to explore cause-and-effect and long-term patterns in project popularity.

- **Random Forest Classification [RQ#3]**: A Random Forest Classifier is used to predict whether a repository is popular based on various factors of project quality. This ensemble method employs multiple decision trees in order to improve prediction and avoid overfitting as compared to individual decision trees. Assumes normality and equal variances, which may not be true for all features. Non-parametric tests (e.g., Wilcoxon) could correct for this but were not employed [14].

- **Feature Importance Analysis [RQ#3]**: Contribution of each repository feature to popularity prediction is extracted from the trained model. The same is employed to identify key factors responsible for the largest success rate of projects. Results are represented using a bar chart ranking top features. MSE is calculated to assess the model's predictive power by measuring the average squared error in actual and predicted popularity labels. Smaller MSE indicates better model precision.

3.5 Data Visualization

- **Time-Series Graph Analysis [RQ#1 and RQ#2]:** Time-series graph analysis is the computation of rolling means of various project metrics over 12 months to remove volatility and identify underlying trends. The measures are graphed against time, separated by the popularity of the projects, to visually explore the relationship between the evolution of network structures and project activity with popularity. This approach was utilized rather than raw trend lines to reduce noise and provide a more accurate representation of metric progress, especially in sparse or irregular data. Nonetheless, using a fixed rolling window size may suppress short-term volatility or delay the detection of abrupt change [13].

- **Box Plot Visualization [RQ#3]:** Box plots are utilized to visualize *pScore* distribution for some project quality metrics. This is to explore trends, outliers, and variability in project popularity based on some repository traits. It provides a precise overview of central tendency, dispersion, and outliers, which is critical for exploratory analysis.

- **Correlation Analysis [RQ#1, RQ#2, and RQ#3]**: Correlation analysis involves the computation of Pearson correlation coefficients for project metrics vs. project popularity. Statistical correlation establishes the relationships' strength and direction, hence one can tell which project dynamic factors are closely correlated with popularity as a project. Pearson correlation was utilized in preference to more complex multivariate methods because of its interpretability, statistical simplicity, and clear definition of linear relationships between single features and popularity. One limitation of this approach is that it can only capture linear relationships and does not provide for interaction between variables or for confounding variables.

4 Results

4.1 RQ#1

Time Series Analysis:

- **Network Complexity:** High social graph density may enable early team contact but compromise communication and reduce project popularity due to rising difficulty in handling over-connections. On the other hand, low technical graph density is associated with higher popularity, which translates to sim-pler technical designs that remain manageable and desirable as projects expand.
- **Collaboration Dynamics:** Successful projects contain more extensive pieces, which mean that strongly connected core groups are important for effective collaboration and decision-making. However, clustering coefficient patterns are found not to be related to popularity, and high-density clustering could reflect local activity but does not necessarily translate to higher overall popularity.
- **Project Activities:** Active participation drives project success. The node count of the developer and project popularity have a high positive correlation, showing the importance of an active developer base. Similarly, higher file node counts reflect active development and operational intensity with high rates of updates driving visibility and keeping project activity levels consistent.

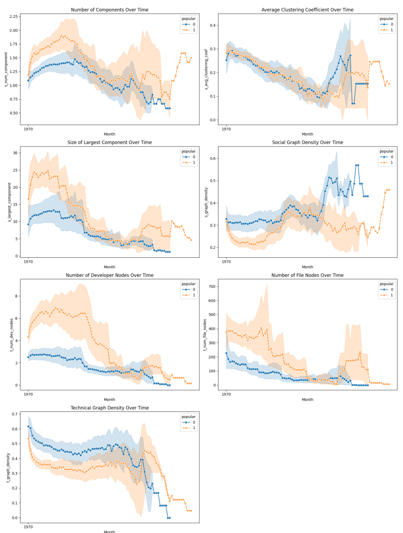


Figure 1: Time series analysis of RQ#1 metrics with popularity

Correlation Analysis:

- **Developer Engagement:** `t_num_dev_nodes_rm` shows the strongest correlation with project popularity ($r = 0.323$, $p < 0.0001$), confirming that greater developer participation is a key factor in project success.
- **File Activity:** `t_num_file_nodes_rm` also shows a strong positive correlation ($r = 0.203$, $p < 0.0001$), indicating that frequent updates and higher file activity increase project visibility and popularity.

- **Network Size:** `s_largest_component_rm` shows a moderate positive correlation with popularity ($r = 0.199$, $p < 0.0001$), suggesting that a strong core network contributes to success.
- **Component Count:** `s_num_component_rm` also shows a positive correlation ($r = 0.180$, $p < 0.0001$), indicating structured network topologies may promote broader community collaboration.
- **Network Density:** Both `s_graph_density_rm` and `t_graph_density_rm` exhibit negative correlations ($r = -0.229$ and $r = -0.240$, respectively, $p < 0.0001$), suggesting that overly dense networks might hinder project scalability and collaboration.
- **Clustering Coefficient:** `s_avg_clustering_coef_rm` shows weak correlation ($r = 0.073$, $p < 0.0001$), implying that local clustering has minimal effect on popularity.

Table 1: Correlation Coefficients between Rolled Metrics and Project Popularity [19]

Feature	**Correlation with Popularity**
s_num_component_rm	0.18
s_avg_clustering_coef_rm	0.07
s_largest_component_rm	0.20
s_graph_density_rm	-0.23
t_num_dev_nodes_rm	0.32
t_num_file_nodes_rm	0.20
t_graph_density_rm	-0.24

4.2 RQ#2

This research question investigates the extent to which developer interaction, collaboration dynamics, engagement, and consistency influence the popularity of open-source projects.

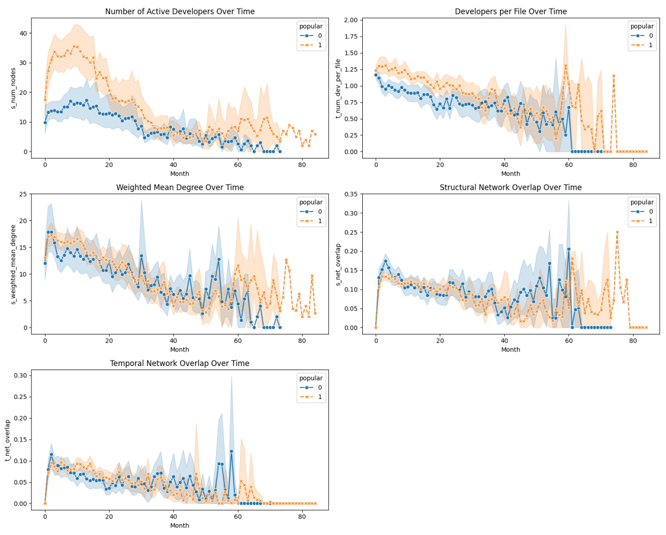


Figure 2: Time Series Analysis of RQ#2 metrics with popularity

Time Series Analysis: A time series analysis was conducted to examine the evolution of developer interaction, collaboration intensity, and engagement consistency across the lifespan of open-source projects. This analysis aimed to identify temporal patterns and differences in key interaction metrics between popular and non-popular projects. From Figure [x], we infer that:

- **Higher Developer Engagement Trends Correlate with Popularity:** The `s_num_nodes` (number of active developers) and `t_num_dev_per_file` (developers per file) consistently show higher values for popular projects during the early and mid phases. This suggests that projects with broader developer participation and collaborative effort tend to gain higher popularity.
- **Stable and Denser Social Interaction Patterns in Popular Projects:** Metrics like `s_weighted_mean_degree` and `s_net_overlap` display smoother and more stable trends for popular projects, indicating more structured and cohesive communication among contributors. In contrast, non-popular projects exhibit more erratic behavior in these metrics, possibly reflecting fragmentation or less coordinated teamwork.
- **Temporal Interaction Consistency Influences Project Visibility:** The `t_net_overlap` (temporal overlap of developers) remains consistently higher in popular projects across most months. This trend suggests that retention of active developers over time contributes positively to project popularity, likely due to sustained contributions, bug-fixing, and feature development.

Results Analysis: From Table 2 [19], we inferred that:

- **Developer Collaboration is the Strongest Predictor of Popularity:** `t_num_dev_per_file` shows the highest correlation ($r = 0.353$, $p < 0.001$), indicating that projects with higher developer collaboration per file tend to be more popular.
- **Developer Engagement Positively Impacts Popularity:** `s_num_nodes` ($r = 0.229$, $p = 0.0023$) confirms that a larger number of active developers is associated with higher popularity, emphasizing the importance of community size.
- **Consistency in Developer Participation Plays a Moderate Role:** `t_net_overlap` has a moderate correlation ($r = 0.177$, $p = 0.0194$), suggesting that sustained developer involvement over time modestly contributes to project popularity.
- **Structural Network Metrics Are Less Influential:** `s_weighted_mean_degree` and `s_net_overlap` exhibit weak or insignificant correlations with popularity ($r = 0.129$ and 0.065 respectively), indicating that structural cohesion alone does not strongly drive popularity.
- **Inter-metric Relationships Are Stronger Than Their Individual Link to Popularity:** High inter-correlations (e.g., `s_weighted_mean_degree` with `s_net_overlap`: $r = 0.74$) suggest these metrics reflect underlying collaboration dynamics, but their influence on popularity is indirect and less impactful than direct engagement and collaboration metrics.

Metric	Correlation with Popularity
s_num_nodes	0.23
t_num_dev_per_file	0.35
s_weighted_mean_degree	0.13
s_net_overlap	0.06
t_net_overlap	0.18

Table 2: Correlation of RQ#2 Metrics with Project Popularity

Through Box plot analysis, we infer that:

- **Developer Engagement:** The median and overall spread of active developers (`s_num_nodes`) is higher for popular projects. Popular projects tend to involve more developers, supporting the idea that broader community participation boosts popularity.

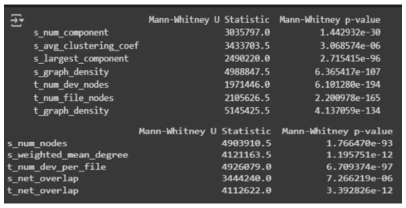
- **Collaboration Intensity:** Popular projects show a higher median and broader upper range for (`t_num_dev_per_file`), suggesting more frequent collaboration per file. This reinforces earlier correlation insights that file-level collaboration is a strong popularity indicator.

- **Developer Connectivity:** There is a slight increase in the median for popular projects for (`s_weighted_mean_degree`), but a significant overlap in the interquartile range (IQR) with non-popular ones. This suggests limited discriminative power, echoing the weak correlation found earlier.

- **Developer Consistency & Retention:** Structural (`s_net_overlap`) and temporal (`t_net_overlap`) overlaps show minimal differentiation between popular and non-popular projects, suggesting that while consistent developer involvement may offer some influence, these metrics alone are not strong indicators of popularity.

4.3 Mann-Whitney Test for RQ#1 & RQ#2

The Mann-Whitney U test was used to establish differential trends among popular and non-popular projects. Popular projects tended to have fewer isolated components, suggesting stronger cohesion and coordination. They also have increased clustering coefficients that suggest localized and close collaborations that are efficient for code optimization. Popular projects were larger in connected components to facilitate easier transfer of knowledge and fewer silo effects. These projects were more socially and technologically graph-dense, representing healthy interactivity conducive to problem-solving and innovation. In particular, successful projects showed more file and developer nodes, representing more complex and thoroughly worked-on codebases. Moreover, higher interaction levels among the contributors were observed, highlighting the importance of positive cooperative dynamics for successful projects.



```
                          Mann-Whitney U Statistic  Mann-Whitney p-value
    s_num_component                      3035797.0          1.442932e-30
    s_avg_clustering_coef                3433703.5          3.068574e-06
    s_largest_component                  2490220.0          2.715415e-96
    s_graph_density                      4988847.5         6.365417e-107
    t_num_dev_nodes                      1971446.0         6.101280e-194
    t_num_file_nodes                     2105626.5         2.200978e-165
    t_graph_density                      5145425.5         4.137059e-134

                        Mann-Whitney U Statistic  Mann-Whitney p-value
s_num_nodes                            4903910.5          1.766470e-93
s_weighted_mean_degree                 4121163.5          1.195751e-12
t_num_dev_per_file                     4926079.0          6.709374e-97
s_net_overlap                          3444240.0          7.266219e-06
t_net_overlap                          4112622.0          3.392826e-12
```

Figure 3: Mann-Whitney U test

4.4 RQ#3

Random Forest Classifier for Feature Importance

- **Regular Reporting is the Single Best Predictor of Popularity:** The feature `reporting_monthly`'s importance score is highest (0.302), which indicates that regularly reported projects tend to be the most popular.

- **Issue Tracking Boosts Popularity:** The feature `has_issue_tracker` ranks second in importance (0.196), suggesting that possessing an issue tracker significantly enhances a project's success by increasing transparency and fostering collaboration.

- **Developer Mailing Lists are Very Important:** Features such as `has_a_commits_mailing_list` (0.110) and `has_a_dev_mailing_list` (0.109) play a crucial role in ensuring continuous communication among developers.

- **Project Documentation and Distribution are Somewhat Important:** Features such as `has_status_file` (0.096), `has_distribution_area` (0.067), and `has_release_signing_keys` (0.059) contribute moderately to popularity, highlighting the importance of structured release and documentation procedures.

- **Website Presence and Code Repositories Contribute Least:** Surprisingly, features such as `has_website` (0.003) and `has_a_code_repo` (0.003) contribute the least to popularity. While these elements are essential, they do not significantly differentiate successful projects.

- **Model Performance:** The Random Forest model gave a Mean Squared Error (MSE) of 0.428, reflecting an intermediate level of prediction accuracy.

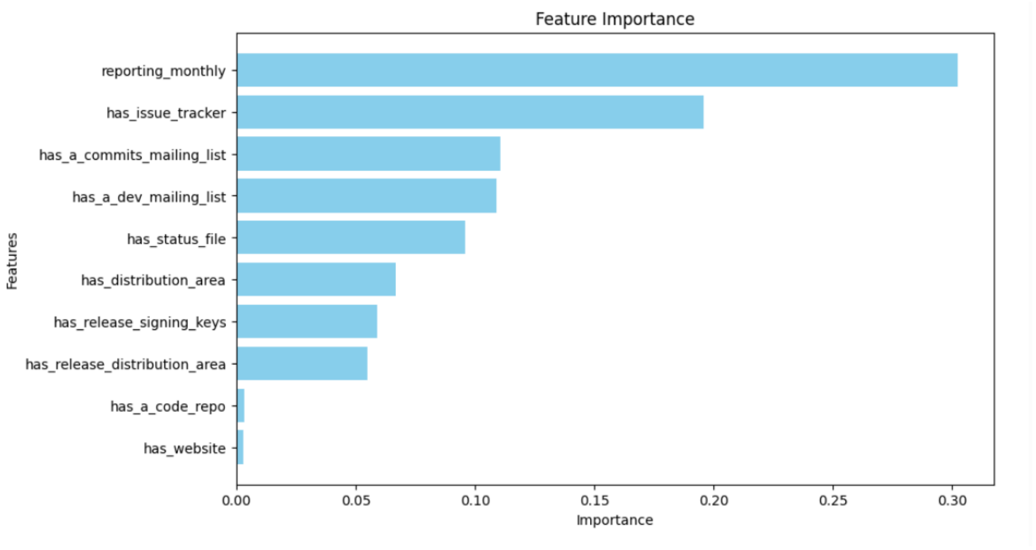


Figure 4: Feature Importance

Feature Correlation with pScore

- **Issue Tracking and Developer Mailing Lists Are Weakly Correlated with Popularity:** `has_issue_tracker` (0.165), `has_a_commits_mailing_list` (0.147), and `has_a_dev_mailing_list` (0.144) are weakly positively correlated with pScore, meaning that projects with issue tracking and developer mailing lists are weakly more popular. However, the correlation is not strong enough to be a definite predictor.

- **Release and Distribution Features Have Minimal Impact:** Characteristics like has_release_distribution_area (0.094), has_release_signing_keys (0.068), and has_distribution_area(0.068) exhibit weak positive correlations with popularity. This indicates that while organized release management may contribute to leading to popularity, it is not a strong influence.

- **Presence of Monthly Reporting and Code Repository Reflects Weak Negative Affiliations:** reporting_monthly (-0.134) represents a weak negative correlation with pScore, suggesting that projects reported monthly are not necessarily more popular. has_a_code_repo (-0.040) shows a close to zero negative correlation, suggesting that having a code repository does not guarantee increased popularity.

Feature	Correlation with pScore	T-Statistic	P-Value
has_issue_tracker	0.165	0.914	0.368
has_a_commits_mailing_list	0.147	0.816	0.421
has_a_dev_mailing_list	0.144	0.798	0.431
has_release_distribution_area	0.094	0.517	0.609
has_release_signing_keys	0.068	0.371	0.713
has_distribution_area	0.068	0.371	0.713
has_status_file	0.064	0.351	0.728
has_website	0.026	0.145	0.886
has_a_code_repo	-0.040	-0.218	0.829
reporting_monthly	-0.134	-0.738	0.466

Table 3: Feature-wise Correlation and T-Test Statistics with *pScore*

T-Test Results for Feature Significance

- **None of the Features Has Strong Statistical Significance:** The p-values of all features are greater than the standard cutoff point (p ¿ 0.05), indicating that none of the features have a statistically different difference in pScore between the projects having the respective feature or not.
- **Weak Implication of Every Single Feature towards Popularity** Features like has_issue_tracker (p = 0.368) and reporting_monthly (p = 0.466) have slightly lower p-values but are not significant enough. This means that although they may have some impact, their impact is not strong enough to be statistically significant.
- **Website Presence and Code Repository Have the Least Influence** has_website (p = 0.886) and has_a_code_repo (p = 0.829)have the largest p-values, corroborating the earlier result that these features are not strongly associated with project popularity.

5 Threats to Validity

Firstly, there's a threat to construct validity i.e., potential defining metrics like download rates or active user count for the measurement of popularity is not considered. The socio-technical and repository quality metrics may not cover all dimensions that define popularity, so there surely a room for unobserved variables. Furthermore, internal validity is also questioned in the context of correlation versus causation issues,i.e., correlations cannot necessarily convey causality. External validity is compromised by a focus on only ASFI projects, limiting generalizability to the open-source world at large [18]. Finally, more well-defined measures could have been taken to define quantile and rolling mean values. Despite these limitations, we utilized multiple analytic strategies and open methodological reporting to build confidence in our findings, encouraging replication and extension to address these concerns.

6 Discussion

6.1 Principal Outcomes

This paper offers an extensive examination of the determinants of open-source project popularity based on socio-technical measurements and repository quality attributes. Our results emphasize the prominent role of active developer interest, vigorous collaboration, formalized network structures, and regular project management practice in achieving project success. The research points out that popularity cannot be just judged upon technical artifacts but a combination of social dynamics, development practices, and project management styles is necessary. Regular reporting, issue tracking, and open collaboration style become key strategies for projects to gain visibility and traction in the open-source community. These findings offer valuable recommendations to open-source development maintainers, contributors, and organizations.

6.2 Comparison to Prior Studies

While several prior studies have attempted to explore the factors influencing the popularity of open-source software (OSS) projects, most of them have examined social and technical dimensions in isolation, rather than adopting a comprehensive socio-technical perspective. Even within technical analyses, the number of metrics assessed has often been limited, and the methodologies largely relied on basic correlation-based evaluations, lacking depth in interpretability and generalizability. Furthermore, temporal dynamics of project evolution have been significantly underexplored. Prior work has seldom investigated how developer interaction patterns and engagement metrics evolve over time and contribute to project popularity. This limitation restricts the ability to understand project behavior longitudinally.

In terms of assessing social aspects, many earlier studies have predominantly relied on survey-based inference, which, while insightful, may be prone to biases and lack objective validation. In contrast, our study leverages network-based social metrics derived from real interaction traces within the Apache Software Foundation Incubator (ASFI) projects, providing more reliable and data-driven insights. Additionally, when evaluating repository quality, prior research has often relied on manually annotated datasets, which may suffer from subjectivity and inconsistency. Our approach, however, incorporates quality indicators directly sourced from the Apache Incubator Clutch dataset, ensuring the accuracy, consistency, and authenticity of project-level metadata. Notably, our findings are partially aligned with existing research of Borges, H. (2022) [6], which identifies the number of contributors as a significant determinant of project popularity. However, such studies often overlook the collaborative and interactional dynamics between developers, which we integrate through socio-technical network metrics. By capturing both the structural and behavioral dimensions of OSS projects, our work presents a more holistic framework for understanding project popularity.

6.3 Case Study: Evaluation of ECharts - Most Popular ASFI Project

We conducted a comparative analysis by evaluating ECharts against the median values of all projects across key social and technical metrics. We did not consider mean values due to the presence of outliers in the data. Our main goal of this case study was to explore if the findings of our research questions corroborate or not.

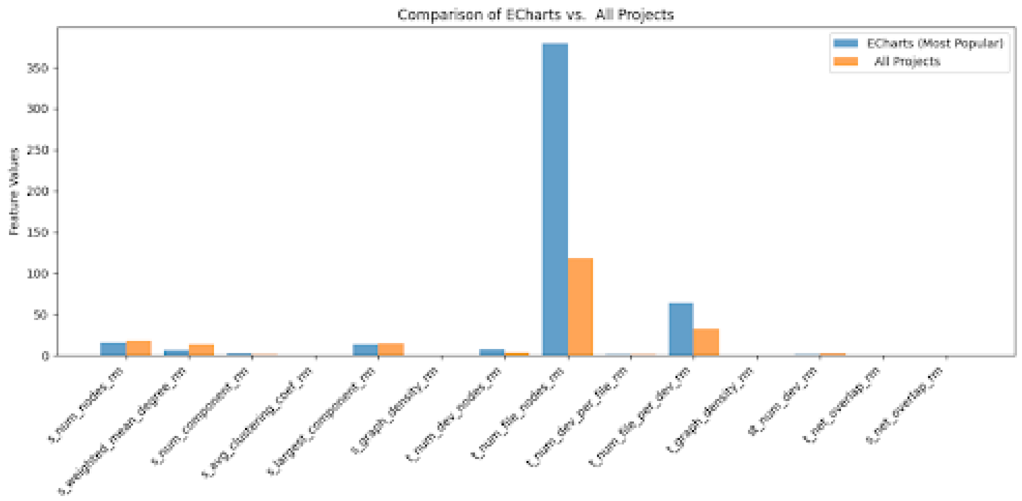


Figure 5: Comparative analysis of ECharts with median project values across key metrics.

Findings: Our analysis revealed that some features exhibit significantly higher values in ECharts compared to the median of projects, suggesting that the considered metrics contribute to its popularity.

ECharts has a significantly higher number of `t_num_file_nodes_rm` (**379.83**) value compared to the median of all projects (**118.33**), indicating a more extensive codebase, better documentation, and a well-structured repository. It also has a significantly higher `t_num_file_per_dev_rm` (**63.65**) value compared to the median (**32.23**), indicating that each developer is responsible for managing a larger number of files. The number of active developers (`t_num_dev_nodes_rm`) for ECharts is **8.0**, which is significantly higher than the median. This suggests that ECharts has a larger and more sustainable developer community, ensuring continuous maintenance, feature updates, and overall project longevity. The largest component size (`s_largest_component_rm`) for ECharts is **14.25**, which is slightly lower than the median value of **15.33**. This indicates that while ECharts has a well-connected structure, it does not significantly exceed the average in terms of the largest connected subnetwork within the project.
Hence, we can conclude that these results support the findings of our research questions.

6.4 Future Work

Future research directions include conducting longitudinal studies to observe how repository quality features [RQ#3] influence project popularity over time. Causality analysis using advanced statistical techniques can help identify relationships between these factors and project popularity. A cross-platform comparison incorporating OSS projects from GitHub, GitLab, and Bitbucket would validate findings across different ecosystems. Additionally, machine learning models could be developed to predict project popularity based on metrics at an early stage, optimizing new projects for popularity. Lastly, qualitative research will be conducted through interviews or surveys with contributors and maintainers to provide deeper insights into the key drivers of project popularity.

6.5 Conclusion

This paper offers an extensive examination of the factors of open-source project popularity based on socio-technical metrics and repository quality attributes. Our results emphasize the prominent role of active developer interest, extensive collaboration, formalized network structures, and regular project management practice in achieving project success. The research points out that popularity is not just a technical artifact but a combination of social dynamics, development practices, and project management styles. Regular reporting, issue tracking, and open collaboration style become key strategies for projects to gain visibility and traction in the open-source community. These findings offer valuable recommendations to open-source development maintainers, contributors, and organizations to assess their project's popularity.

7 Team Membership and Attestation

All team members listed below consent to the content of this report, affirm the originality of this work and commit to upholding academic integrity throughout the project. Each member's contribution is mentioned below:

- *Shaik Haseeb Ur Rahman*: Contributed to the literature review and background work; performed end-to-end data preprocessing of both datasets and worked on statistical analysis and visualizations for RQ#2 and RQ#3; contributed to the final inference formulation and limitation analysis.

- *Aayusha Hadke*: Contributed to the literature review and background work; developed the methodology for data collection and preprocessing for RQ#1, worked on statistical analysis for RQ#3, and contributed to the results and conclusion sections.

- *Celine John Philip*: Performed data preprocessing for RQ#2 and RQ#3; contributed to visualizations, and worked on statistical analysis and interpretation of results for all three research questions.

- *Aditi Agrawal*: Contributed to the literature review and background work; worked on the development of visualizations for all research questions and contributed to writing the results section.
- *Krishna Karthik*: Performed end-to-end data preprocessing for RQ#2; worked on the implementation of the Mann-Whitney test, Random Forest Classifier, and other statistical methods; contributed to the discussion and conclusion sections.

Code & Data Availability Statement

The adopted dataset of RQ#1 and RQ#2 can be accessed at [4], and RQ#3 can be accessed through [5]. This study's final code and datasets can be accessed here, Github Repo - SE Project.

References

[1] Yin, L., Chen, Z., Xuan, Q., & Filkov, V. (2021). *Sustainability forecasting for Apache incubator projects*. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021) (pp. 1056–1067). Association for Computing Machinery. https://doi.org/10.1145/3468264.3468563

[2] Yin, L., Zhang, Z., Xuan, Q., & Filkov, V. (2021). *Apache Software Foundation Incubator Project Sustainability Dataset*. In Proceedings of the IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) (pp. 595–599). https://doi.org/10.1109/MSR52588.2021.00081

[3] Yin, L., Zhang, X., & Filkov, V. (2023). *On the self-governance and episodic changes in Apache incubator projects: An empirical study*. In Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia (pp. 678–689). IEEE. https://doi.org/10.1109/ICSE48619.2023.00066

[4] Anonymous, A. (2024). *Forecasting Sustainability of Apache and Eclipse Incubator Projects*. Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.14499305`

[5] Apache Software Foundation. (2024). *Apache Incubator Clutch*. Retrieved from `https://incubator.apache.org/clutch/`

[6] Borges, H., Hora, A., & Valente, M. T. (2016). *Understanding the factors that impact the popularity of GitHub repositories*. In Proceedings of the 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), Raleigh, NC, USA (pp. 334–344). IEEE. https://doi.org/10.1109/ICSME.2016.31

[7] Alsmadi, I., & Alazzam, I. (2017). *Software attributes that impact popularity*. In Proceedings of the 2017 International Conference on Information Technology (ICITech). https://doi.org/10.1109/ICITECH.2017.8080001

[8] Aggarwal, K., Hindle, A., & Stroulia, E. (2014). *Co-evolution of project documentation and popularity within GitHub*. In Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014) (pp. 360–363). Association for Computing Machinery. https://doi.org/10.1145/2597073.2597120

[9] Saini, M., Verma, R., Singh, A., & Chahal, K. (2020). *Investigating diversity and impact of the popularity metrics for ranking software packages. Journal of Software: Evolution and Process, 32*, e2265. https://doi.org/10.1002/smr.2265

[10] Eisty, N. U., Thiruvathukal, G. K., & Carver, J. C. (2018). *A survey of software metric use in research software development.* In *2018 IEEE 14th International Conference on e-Science (e-Science)* (pp. 212–222). IEEE. https://doi.org/10.1109/eScience.2018.00036

[11] Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012). *Social coding in GitHub: Transparency and collaboration in an open software repository.* In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12) (pp. 1277–1286). Association for Computing Machinery. https://doi.org/10.1145/2145204.2145396

[12] Weber, S., & Luo, J. (2014). *What makes an open source code popular on GitHub?* In Proceedings of the 2014 IEEE International Conference on Data Mining Workshop (pp. 851–855). IEEE. https://doi.org/10.1109/ICDMW.2014.55

[13] Yang, J., & Leskovec, J. (2011). *Patterns of temporal variation in online media.* In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM '11) (pp. 177–186). Association for Computing Machinery. https://doi.org/10.1145/1935826.1935863

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

[15] Corral, L., & Fronza, I. (2015). *Better code for better apps: A study on source code quality and market success of Android applications.* In Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems (pp. 22–32). IEEE. https://doi.org/10.1109/MobileSoft.2015.10

[16] Zhu, J., Zhou, M., & Mockus, A. (2014). *Patterns of folder use and project popularity: A case study of GitHub repositories.* In Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14), Article 30 (pp. 1–4). Association for Computing Machinery. https://doi.org/10.1145/2652524.2652564

[17] Capra, E., Francalanci, C., Merlo, F., & Rossi-Lamastra, C. (2011). *Firms' involvement in open source projects: A trade-off between software structural quality and popularity.* Journal of Systems and Software, 84(1), 144–161. https://doi.org/10.1016/j.jss.2010.09.004

[18] Sajnani, H., Saini, V., Ossher, J., & Lopes, C. V. (2014). *Is popularity a measure of quality? An analysis of Maven components.* In Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 231–240). IEEE. https://doi.org/10.1109/ICSME.2014.45

[19] OpenAI. (2024). ChatGPT (GPT-4-turbo) [large language model]. Retrieved from `https://chat.openai.com`. *Used during the report drafting process for table creation and indentation purposes.*

[20] Grammarly Inc. (2024). Grammarly AI Writing Assistant [software tool]. Retrieved from `https://www.grammarly.com`. *Used during the report drafting process for the spell-check process.*