university of groningen

faculty of science and engineering

The Influence of Ecosystem-Wide Experience and Collaboration on Pull Request Acceptance in Open-Source Software Ecosystems

Willem Meijer

university of groningen

faculty of science and engineering

University of Groningen

The Influence of Ecosystem-Wide Experience and Collaboration on Pull Request Acceptance in Open-Source Software Ecosystems

Master's Thesis

To fulfil the requirements for the degree of
Master of Science in Computing Science – Software Engineering and Distributed Systems
at the University of Groningen under the supervision of
Asst. prof. dr. A. Rastogi (Bernoulli Institute, University of Groningen)
and
Asst. prof. dr. M. Riveni (Bernoulli Institute, University of Groningen)

Willem Meijer (s4509412)

August 30, 2023

Acknowledgements

I don't think a document like this should dare to present itself without a proper *"thank you"* to everyone who helped me set up and execute this project.

Thank you, Ayushi, for the endless support, feedback, thoughts, talks, and the various walks we have had throughout this project. I don't know what this project would've looked like if you weren't involved, but it likely wouldn't have reached the same results, and it most definitely wouldn't have been as much fun. I couldn't have wished for a better supervisor.

Thank you as well, Mirela, for your help throughout the project. There were about a thousand things I wouldn't have known about social network analysis if you hadn't pointed me in the right direction.

Thank you to the other professors and peers, both at the University of Groningen and outside, who inspired me both before and during the execution of this project. Thanks to the technical staff who gave me access to the resources necessary to complete this work, as I would've been completely lost without them.

Thanks to the Computing Science programme and all of the staff in it. I genuinely appreciate the effort you put in and the opportunities you have given me. Following the programme was an absolute blast. I am genuinely a little sad to think it's over after this.

Thanks to everyone. To those who joined my life. To those who stayed in my life. To those who left my life. Thank you. Thank you. Thank you. May our paths cross again someday.

– Willem

Samenvatting

Het pull-based ontwikkelingsmodel ligt ten grondslag aan wereldwijde samenwerking in open-source softwareontwikkeling. Dit model onderscheidt zichzelf door de beslissingen die over software worden gemaakt los te koppelen van softwareontwikkeling. Dit wordt mogelijk gemaakt door ontwikkelaars aanpassingen te laten maken in een lokale versie van de code. Dit geeft hen de mogelijkheid om een voorstel te doen deze te integreren in de publieke versie door middel van pull requests (aanpassingsverzoeken). Deze methode wordt vaak gecombineerd met issue-beheer systemen. Gebruikers en ontwikkelaars kunnen hier *"issues"* plaatsen om nieuwe ideeën te presenteren, vragen te stellen of problemen te beschrijven.

Onderzoek dat in de afgelopen tien jaar is uitgevoerd heeft verscheidene factoren gevonden die beïnvloeden of een pull request wordt geaccepteerd of afgewezen. Veel hiervan zijn gemeten binnen één project. Moderne software bestaat echter zelden nog als solitaire projecten omdat ze deel zijn geworden van grotere software ecosystemen.

Omdat er weinig onderzoek is gedaan naar het belang van software ecosystemen in relatie tot pull-based softwareontwikkeling, getuigt deze these voor de relevantie door het effect op pull request acceptatie te testen. Dit is gedaan door de impact te testen van opgedane werkervaring binnen een ecosysteem, opgedane werkervaring binnen projecten met een onderlinge technische afhankelijkheid, en directe en indirecte samenwerking waar een ontwikkelaar aan deel heeft genomen. Dit onderzoek benadrukt dit effect op een speciale groep ontwikkelaars: nieuwkomers. Dit zijn ontwikkelaars die voor het eerst een pull request indienen binnen een project.

Om deze studie uit te voeren, is een dataset van ongeveer 1.8 miljoen pull requests en 2.1 miljoen issues verspreid over 20.052 projecten, verzameld uit het NPM ecosysteem — een open-source software ecosysteem voor JavaScript projecten. Deze studie gebruikt een combinatie van sociale netwerkanalyse, gemengde logistieke regressie en random forest om de impact en de voorspellende kracht van de gemeten variabelen te testen.

De resultaten van dit onderzoek tonen aan dat alle ecosysteem-brede factoren pull request acceptatie positief beïnvloeden. Dit suggereert dat opgedane werkervaring binnen het ecosysteem door actieve deelname in issue-tracking systemen, het indienen van pull requests, en samenwerking met de integreerder van de pull request en ervaren ontwikkelaars een positief effect heeft op open-source ontwikkelaars. Dit geldt met name voor nieuwe ontwikkelaars. De resultaten van dit onderzoek suggereren dat het voorspellen of een pull request die ingediend is door een nieuwkomer lastig is, gezien slechts een F1 score van 0.83 werd behaald. Het complete model bereikte wel een goede score van 0.92.

Vervolgonderzoek kan uitbreiden op dit werk door het te repliceren in andere ecosystemen (bijvoorbeeld PyPI of OpenStack), te beschrijven waarom dit effect bestaat, en te beschrijven hoe dit optimaal ingezet kan worden binnen projecten.

Abstract

The pull-based development model is one of the fundamental methods used for global collaboration in open-source software engineering. This model defines itself by separating software decision-making from software development. This is possible because it allows individual developers to make changes on a separate branch of the program. Then, they can propose to include these changes in the publicly available code by submitting pull requests (change requests). This method is frequently combined with issue-tracking systems. Users and developers can use this to post *"issues"* which contain new ideas, describe problems, or ask questions.

Research performed in the last decade identified various factors that affect whether a pull request is accepted or rejected, many of which are tracked within a single project. However, modern-day software rarely exists as solitary entities as they have grown to become part of a larger software ecosystem.

Because little prior literature studied the importance of software ecosystems with respect to pull-based development, this thesis vouches for its relevance by testing its relationship with pull request acceptance. This is done by measuring the impact of experience acquired in an ecosystem, experience acquired in projects involved in a technical dependency, and the number of direct and indirect collaborations a developer has been part of. This thesis highlights one special case: first-time contributors. These are contributors who make their first technical contribution to a project.

To conduct this study, a dataset of approximately 1.8 million pull requests and 2.1 million issues was collected, spanning 20.052 projects sampled from the NPM ecosystem — an open-source software ecosystem for JavaScript projects. This study employed a combination of social network analysis, mixed effects logistic regression and random forest to measure the impact and predictive strength of identified features.

The results showed that ecosystem-wide factors uniformly positively influence pull request acceptance. This suggests that gaining experience within the ecosystem through active participation in issue-tracking systems, submitting pull requests, and collaborating with integrators and experienced developers benefit all open-source contributors. This is especially important for first-time contributors. The results suggest that predicting the outcome of a pull request submitted by a first-time contributor is more difficult as it yielded only an F1 score of 0.83. However, the overall classification performance reached a good score of 0.92.

Future work could expand on the suggestions of this study by observing the same phenomena in other ecosystems (e.g., PyPI or OpenStack), by identifying why this effect occurs, and how projects can harness it.

Keywords: *open-source software ecosystem, mining software repositories, social coding platform, software dependencies, first-time contributors.*

Contents

List of Figures

List of Tables

1 Introduction

Pull-based development [1], [2] is one of the fundamental processes used for global collaboration in open-source software engineering. The strength of this method is that it separates software decision-making from software development. It allows developers to create a separate *"branch"* of the software that can be changed without affecting the public version. Once these changes can be integrated into the public version, the developer can propose this through a *"pull request."* Typically, this pull request is then reviewed by a core project member, who can choose to apply the changes to the code, decline them altogether, or request some changes. Open-source software teams commonly combine pull-based development with issue-tracking systems [3]. These systems track *"issues,"* containing new ideas, bug reports, and questions.

Research done in the past decade has explored the dynamics of the pull-based development model, describing topics like pull request decision-making [4] or the time it takes to reach these decisions [5]. Most of these studies addressed *"intra-project factors"* of pull request acceptance, referring to things that can be measured within a single project, like the number of comments a pull request has [4], [6], [7], or the number of previously accepted pull requests someone has submitted [4], [8]–[10].

While several studies included factors that reach beyond the borders of a single project [4], [11]–[14], only the work by Dey *et al.* [15] studied the effect of ecosystem-wide factors of pull request acceptance. An open-source software ecosystem can be defined as *"a network of open-source software communities working on a common technology"* [16], [17]. An example is the NPM ecosystem, which contains millions of open-source software JavaScript packages built to be reused in other software. In their work, Dey *et al.* [15] concluded that pull requests submitted by developers with ecosystem-wide experience are more likely to be accepted than those by people who do not.

1.1 Research Questions

The starting point of this study is the work of Dey *et al.* [15], who suggested the importance of ecosystem-wide experience in pull-based development. Consequently, the main question of this study is very similar to theirs:

"Do ecosystem-wide experience and collaboration influence pull request acceptance decisions in another project within the software ecosystem?"

To answer this research question, this study first attempts to replicate the results of Dey *et al.* [15] using an exact independent replication study [18]. Their study identified a relationship between pull request acceptance ecosystem-wide experience acquired through pull requests and code commits. Although pull requests and code commits have been studied in relationship with pull request acceptance [4], experience acquired by submitting issues by commenting on other people's issues/pull requests has not been studied. This is a notable gap in the literature, as research has shown that developers commonly employ non-developer roles before making a contribution [19]. Therefore, this thesis includes experience acquired in issue-tracking systems and by commenting on issues and pull requests as both are non-contributor activities. Consequently, the first research question is:

RQ1 *Does prior experience (including non-code contributions) within the ecosystem influence the chances of developer pull request acceptance in another project within the ecosystem?*

A second topic of interest is the relevance of dependencies between projects inside the ecosystem. Dey *et al.* [15] studied this factor to some extent by including whether a pull request submitter contributed to any dependent project and found that this positively affects pull request acceptance. Although this is a highly relevant find, their work did not consider the direction of a dependency (projects can have incoming and outgoing dependencies) by only including incoming dependencies. Therefore, the second research question is:

RQ2 *Does prior experience in projects involved in a technical dependency influence developer pull request acceptance in the projects they have a technical dependency with?*

In the past, various studies observed the impact of *"social distance"* between developers (measured using GitHub's follower feature), suggesting that changes proposed by developers who are closely connected to their pull request integrator [11], [12] and developers who are well-connected inside GitHub [13], [14] have a higher chance of being accepted.

These studies measured the social distance between developers using GitHub's follower feature. Although this feature provides insights into the social component of software engineering, it might not accurately represent professional connections as there are no restrictions on who can follow whom. To test the effect of professional connectedness, this study includes two new factors: direct and indirect collaboration between developers. Respectively, these address the number of collaborations between two developers and the quality of their collaboration. Therefore, the third research question is:

RQ3 *Does collaboration within the ecosystem influence the chances of developer pull request acceptance in another project within the ecosystem?*

Finally, this thesis tests the effect of all aforementioned factors with a special type of developer: first-time contributors. First-time contributors are developers who have never made a technical contribution to a project. First-time contributors have indicated that lack of technical expertise is a barrier when contributing to a new project [20]. Consequently, Rastogi *et al.* [21] hypothesised that newcomers who acquired experience within the same ecosystem might suffer less from this problem. However, they have not evaluated this expectation yet. Therefore, to evaluate their hypothesis, the final research question of this thesis is:

RQ4 *Does prior experience within the ecosystem influence the chances of first-time contributor pull request acceptance in another project within the ecosystem?*

1.2 Thesis Outline

The rest of this thesis is organised as follows. Section 2 introduces the necessary background information and several related studies. The methodology is described in Section 3, elaborating on the data collection process, metrics, and data analysis methods. The results are presented in Section 4, of which the implications are described in Section 5. Finally, Section 6 concludes this work and describes potential future research directions.

2 Theoretical Framework

The following section draws the theoretical framework of this work. Section 2.1 introduces the necessary literature describing open-source software ecosystems, the pull-based development model, and first-time contributors. Then, section 2.2 introduces several recent studies directly related to this study.

2.1 Background

The open-source concept is designed with the premise that it is publically accessible, allowing anyone to examine and modify it according to their requirements. One widely used method for collaboration among open-source communities is the pull-based development model [1], [2]. The pull-based development model separates decision-making and software development. In this model, developers can create a local copy of the code and propose changes to be *"merged"* into the public version by submitting a *"pull request."* One advantage of this development process is that although anyone can make changes to software, not every change is immediately applied to the public version of the code.

Open-source development commonly uses issue-tracking systems as well [3]. In this system, users and developers can post *"issues"* posing new ideas, submit bug reports, or ask questions in a blog-like fashion. Issues clearly distinguish themselves from pull requests as they do not require writing any code, making them much more accessible to the general public and enabling gaining experience with the project without making any code contributions. Code hosting platforms like GitHub facilitate pull-based development and issue-tracking systems to simplify software development, making it relatively accessible to software developers and users.

An open-source software ecosystem refers to *"a network of open-source software communities working on a common technology"* [16], [17]. These ecosystems are crucial as very few open-source software exists in complete isolation because these systems very frequently interact and depend on other projects [22]. An example is the NPM package library, a package repository for JavaScript software, which contains over two million projects at the time of writing. In this ecosystem, most packages reuse the functionalities in another project, creating a technical dependency between them. For example, a web application can be built using the React framework, reusing its functionality (e.g., the functionality necessary to make a button) and expanding it wherever necessary.

Software ecosystems make it possible to transfer the knowledge acquired in one project to another. People who transfer this knowledge are called knowledge brokers, and previous studies have reported their influence on project downloads [23]–[25]. These studies suggest that participating in multiple projects can be beneficial and harmful depending on the level of involvement of the contributor [23]–[25].

Most studies addressing pull-based development have reported that development experience within singular projects [1], [8]–[12], [26]–[29] is beneficial for the productivity of continued development in it. Consequently, it will be interesting to see how the experience acquired in one project translates to another, which is addressed in the first research question.

One type of knowledge broker is a first-time contributor with experience inside the software ecosystem. First-time contributors take their ecosystem-related knowledge and bring it to the new project. Many open-source software communities depend on the arrival of newcomers as, at face value, the

threshold of leaving and joining a project is relatively low [30]. Previous work has identified various reasons why first-time contributors choose to discontinue cooperation even though they might have been motivated to contribute at the start [20]. They found that the changes proposed by newcomers are more frequently refused [7], [26], [29], [31].

One of the barriers to participation mentioned in the literature is the lack of technical expertise, as some tasks are too complicated for newcomers. At the same time, integrators have said to evaluate newcomers on their technical merit [32]. In contrast, experience is shown to speed up the onboarding process of recruits in private companies [33], and another study hypothesize that prior ecosystem experience matters [21]. This raises two questions: 1) do newcomers with expertise within the same ecosystem perform better? Moreover, 2) do newcomers who have cooperated with integrators and had a chance to prove their technical merit perform better? Both of these can be answered by addressing RQ4 of this study in tandem with the other three research questions.

2.2 Related Work

2.2.1 Factors influencing pull request decisions

Recently, Zhang *et al.* [4] performed a systematic literature review on pull request decision-making, identifying 94 factors of pull request acceptance, which they tested across different scenarios (e.g., the general case vs. self-integrated pull requests). Most of these are intra-project factors (i.e., factors within a single project). An example of this is the number of pull requests submitted in a project [1], [4], [8]–[10]. Reading the work of Zhang *et al.* [4] is recommended for a complete overview of these

Name	Description
PR commit count	The number of commits in the pull request [1], [7], [9], [10], [13], [26], [34], which Zhang *et al.* [4] found has a positive impact on pull request acceptance, however, its effect is inconsistent throughout the literature.
PR age in minutes	The time between the pull request was opened and when it was closed [10], [26], [35], which has a negative impact on pull request acceptance.
Integrator experience	The experience of the pull request integrator in terms of the number of integrated pull requests at an intra-project level [36], having a positive impact.
PR is self-integrated	Whether the pull request submitter and integrator are the same contributors [4], which they found has a negative impact.
PR has comments	Whether the pull request has comments [6], [7], having a negative impact.
PR has com. ext. contr.	Whether someone who is not a contributor to the project nor is the submitter or integrator commenting on the pull request [6], which has a positive impact.
PR text contains "#"	Whether the title or description of the pull request contains a *"#"* [1], [13], which in GitHub is used to reference another issue or pull request and has a positive impact.
First-time contributor	Whether the submitter of the pull request has successfully contributed to the project before [7], [26], [29], [31], which generally has a negative impact. Although this feature is not part of the recommended list of control variables, it is added as it is an integral variable to this study (as described in section 1).

Table 1: Overview of control variables suggested by Zhang *et al.* [4] used in this study.

factors. However, the control variables used in this paper are listed in Table 1.

Although most existing factors explaining pull request decisions are technical, various studies have identified the importance of human and social factors for pull request acceptance. For example, the work of Yu *et al.* [13] identified the fraction of core developers who interacted within the past month as a proxy of the social strength between the core developers and the project. The study shows that social strength positively influences pull request acceptance [13]. Another work of Rastogi *et al.* [8], [37] identified a relationship between the geographical location of a contribution and pull request acceptance, indicating a disparity in pull request decision-making depending on the region. Finally, the work of Terrell *et al.* [38] identified a relationship between a contributor's gender and pull request acceptance.

2.2.2 Project-transcending and Ecosystem-Wide Factors

Some prior studies have explored the effect of factors that transcend a single project on pull request acceptance [4], [11]–[14], [39]. Most of these [4], [11]–[14] measured the effect of the follower feature in GitHub on pull request acceptance, using this as a proxy for social connectedness within the platform. These studies found that general popularity (measured using the number of accounts someone follows and how often they are followed) positively affects pull request acceptance [13], [14]. Similarly, following the integrator of your pull request was found to have a positive impact too [11], [12].

Celińska [39] studied the effect of various social factors on the number of pull requests and forks a developer receives. They included the number of comments placed on the repositories of other people and the number of issues submitted in their repositories. They found that this positively affects project forks and pull requests received, suggesting the positive influence of ecosystem-wide collaboration. This study expands on these studies by asking RQ3.

Finally, the work of Dey *et al.* [15] lies closest to ours in addressing the effect of ecosystem experience on pull request acceptance. They consider three ecosystem factors: the submitter's pull request track record (measured as the submitted pull request count and acceptance rate), their general experience (via commits in the ecosystem and number of projects worked on), and whether they have worked on a dependent project before. The study showed that the pull request acceptance rate, code commits, projects worked on, and experience in a dependent project increases the chances of pull request acceptance and the total number of pull requests submitted decreases it. They identified several non-monotonic elements in the relationship between ecosystem-wide experience and pull request acceptance by evaluating the partial dependence plots of their models (partial dependence plots indicate how the model's prediction changes when a single variable is changed) However, they largely attributed this to bots as they were overwhelmingly present in these sections of their dataset. This thesis expands on their results by asking RQ1 and RQ2.

3 Methodology

The following section describes the methodology of this work. Section 3.1 describes the data collection process, identifying the used data sources and the criteria used to sample these. This is followed by Section 3.2 that introduces the metrics used in this study to represent developer experience and collaboration. Finally, Section 3.3 introduces the predictive and statistical methods used to answer the posed research questions.

3.1 Data Collection

Three types of data are required to answer the posed research questions: pull requests, issues and an index of project dependencies from a software ecosystem. Although various open-source projects use media like mailing lists, this study limits itself to issues and pull requests as these are integrated into popular social coding platforms like GitHub or GitLab, whereas other media are not. A consequence is that there is no systematic means of testing whether projects use these media and if they do, where the relevant archival data for this is kept.

The starting point of this research is the dataset created by Katz [40] that contains project archive data of 32 software package ecosystems (including platforms like NPM, Maven, Go, and PyPI). The data contains information like package names, descriptions of repository data, and technical dependencies. Although Katz's dataset only contains data up to 2020, thus missing recent data, it is still valuable as creating a similar dataset from scratch is very time-consuming. Even though Katz's dataset contains data on many different ecosystems, this study focuses on the NPM ecosystem, a JavaScript package library. The NPM ecosystem is chosen for three reasons: because Dey *et al.* [15] used this ecosystem as well, for which results can be directly compared, 2) because it is the largest ecosystem in the dataset, and 3) because NPM packages have explicit dependencies on each other.

Because the dataset of Katz [40] does not contain any information about pull requests and issues, this data is collected separately. However, to facilitate this, their dataset needs to be sampled as it contains a total of 1.2 million NPM packages which is too large. This is done in two stages: 1) collecting pull request and issue data of a set of popular projects, and 2) collecting this data from projects that have an incoming/outgoing dependency on any of the popular projects.[1] This process is shown in Figure 1[2] (this diagram is elaborated in Section 3.1.3). This second set of projects is added to ensure that the dataset contains projects with a technical dependency on another project in the dataset, making it possible to answer RQ2.

3.1.1 Project Selection

To simplify the data collection and analysis processes, a simple set of inclusion criteria was defined that the selected projects had to meet: 1) they must have a working GitHub repository URL in Katz [40]'s dataset, 2) they must not be a fork of another project, and 3) the project must have at least five valid pull requests to ensure that the project uses pull requests systematically (*"validity"* is

[1] Note: to improve the validity of the study, this last step was performed a second time. However, the process finished too late, and thus this data was ultimately not included in this study.

[2] Although the diagram makes it seem that the data collection was highly structured, the process was much more iterative. Following the steps of this diagram would (should), however, yield the same results as those presented in this thesis.

elaborated in Section 3.1.2). Although some NPM projects use platforms like GitLab or Bitbucket, their total count was lower by two orders of magnitude. Therefore, these were excluded from this study.

In addition, popular projects were sampled using an additional criterion: *"the project must have at least 10,000 downloads in the past 16 months."* This definition aligns with the work of Dey *et al.* [15] and acts as a proxy measurement for project activity, assuming that more popular projects have more development activities.

Similarly, dependency projects were sampled with one additional criterion: *"the project must be an incoming or outgoing dependency of a popular project."* This ensures that the experience acquired in projects involved in a technical dependency is collected. If only popular projects were considered, this would not be fully possible as these projects might be unrelated.

Although the number of projects that popular projects had an outgoing dependency on was within the scope of this research, the number of projects that popular projects had an incoming dependency on went well beyond that scope.[1] Therefore, this last set was randomly sampled to match the number of outgoing dependent projects. However, as it became apparent that these projects use about half as many pull requests, the sample was increased to approximately twice its size.[2]

3.1.2 Development Activity Selection

The previous paragraph introduced the notion of validity for pull requests. Development activities (i.e., pull requests and issues) can be invalid for many different reasons, for which they are removed from the dataset. To filter these, a series of four inclusion criteria were defined:

- *They cannot be submitted by a deleted account.* When users delete their account, their development activity data are assigned to a *"ghost"* user. Because the ghost user has a substantial number of activities assigned to it, it might skew the results.
- *The development activity is closed.* This is done to ensure the discussion in the activity has come to fruition.
- *The development activity has no missing data.* In some cases, development activities have missing data (e.g., missing user details), which cannot be used for inference.
- *They cannot be submitted by a bot.* Because some of the observations measured by Dey *et al.* [15] were attributed to the presence of bots, these are explicitly left out in this study. Many actions in GitHub can be automated using bots (e.g., *"Dependabot"*, which can manage dependency updates and send notifications), which, when unaccounted for can affect research outcomes when studying developer behaviour as bots create a substantial number of development activities.

Although the first three criteria are relatively easy to apply, dealing with bots is not. GitHub metadata was used as a first line of defence, as bots can be marked by their creators. Although many bots could be filtered in this manner, various bots use user accounts for which this method does not work

[1] The number of new projects with an incoming dependency from the popular projects was 12.566 whereas, the number of projects with an outgoing dependency was 444.803, which is approximately a third of the dataset created by Katz [40].

[2] From 12.556 to 26.376 projects.

perfectly. Luckily, bot detection in social coding platforms is an active field of research [41]–[44], for which there are various solutions to detect bots using different data sources. None of these classifiers could be directly applied to this because of stringent data requirements and the amount of manual evaluation or classification required to use them. However, the classifications of Dey *et al.* [41], [45] and Golzadeh *et al.* [44], [46] are publicly available. Therefore, the accounts identified in these studies were removed as well.

Since neither of these methods is perfect either (by nature, but also because neither is very recent), this study expands on these methods even further. To ensure no major bots are missed, all users with over 400 closed pull requests (slightly less than 500 users) have been manually checked and added to a separate blacklist, excluding four additional bots. To determine whether a user is a bot, their GitHub account is visited, and if anything on it suggests it is a bot, it is marked. For each user marked as a bot, it is very obvious that they are. An example of this is *"Mrs Flux"*[1] whose account description is *"I'm a bot!"* Frequently, a minimal amount of information is shared on a user account. This makes it impossible to identify them as either a bot or a real developer. In these cases, minimising false-positive classifications was given priority, for which these accounts are kept in the dataset.

3.1.3 Data Collection Process

The data used in this study is composed of different sources: libraries.io [47], NPM,[2] and GitHub[3] which is queried using GrimoireLab [48] and the public GitHub API (wherever GrimoireLab could not reach). The basic process consists of three stages: 1) collecting popular projects, 2) collecting dependency projects, and 3) merging and pre-processing the data (shown in Figure 1).

To identify popular projects, the popularity of each project in the Katz [40] is collected using the NPM API. If these projects met the popularity criterion described in Section 3.1.1, they were kept, resulting in a list of 25.545 projects. This is then followed by collecting their respective pull request data. After, any project with fewer than five pull requests was removed.[4] Then, issues were collected using the resulting project list. Finally, the inclusion criteria described in Section 3.1.1 and Section 3.1.2 were applied.

Using the pre-filtered project list, dependency projects were sampled using more or less the same process. The major difference is that projects are sampled using incoming and outgoing dependencies of the previously collected projects. Although the number of outgoing dependencies was sufficiently small to collect entirely, the number of incoming dependencies went well beyond that. Therefore, this set was randomly sampled. Initially, the sample size was equal to that of the outgoing dependencies. However, it became apparent that this group contains about 53% fewer pull requests, for which the sample was increased. Finally, the pull request and issue data were collected and filtered using the defined inclusion criteria.

[1] See Mrs Flux at github.com/mrsflux.

[2] See the NPM website: https://www.npmjs.com/.

[3] See the GitHub website: https://github.com/.

[4] The primary reason for this was to minimise the time required to collect all the issue data (as collecting pull request data for popular projects took approximately two weeks of continuous querying), and, by definition, these projects could not meet the required minimum number of *"valid"* pull requests (described in Section 3.1.1 and Section 3.1.2).

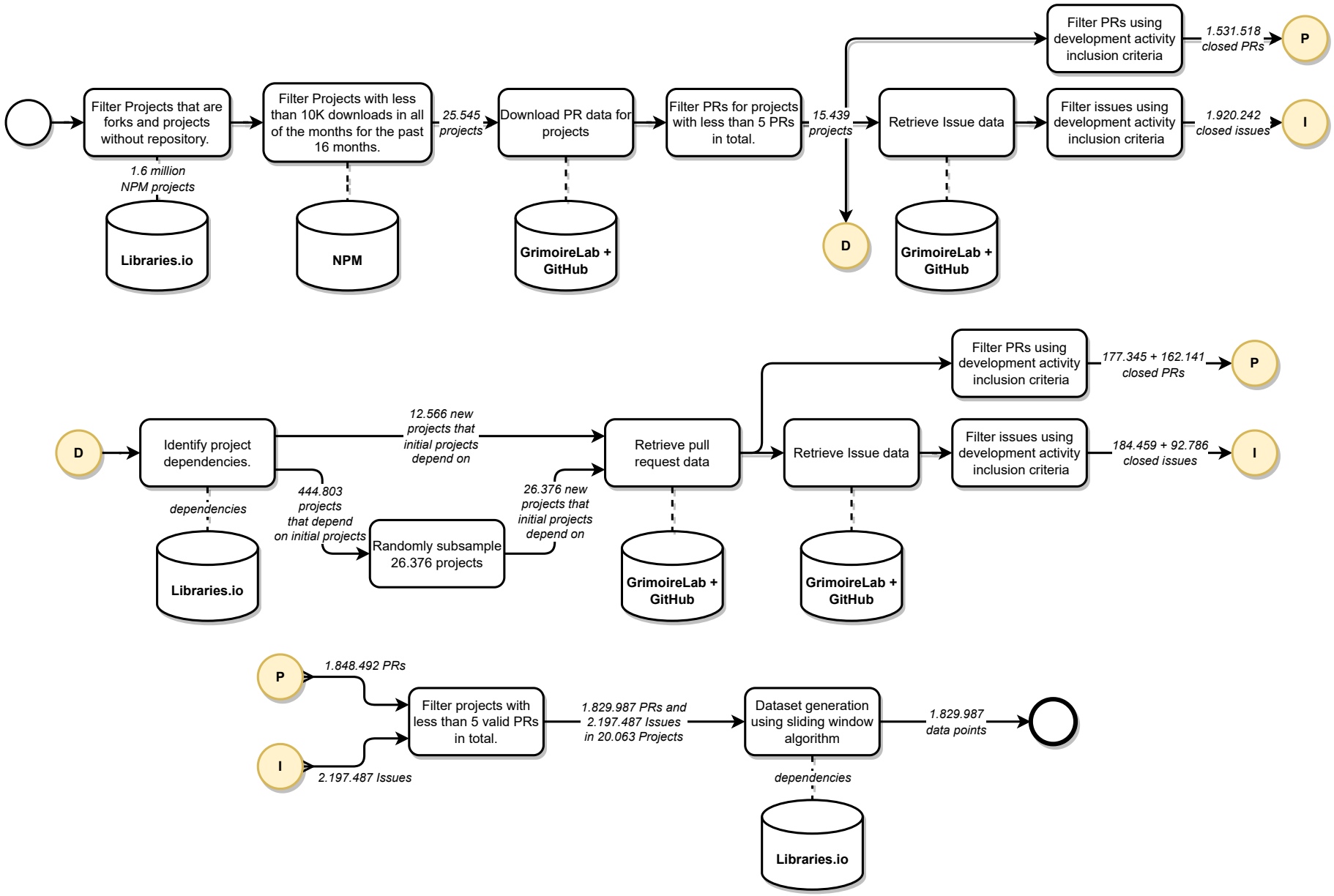


Figure 1: Overview of the data collection process. The top shows data collection of core projects, the middle shows data collection of periphery projects, and the bottom shows the data aggregation process.

An observant reader might notice that at point *D* of the data collection process, not all of the defined inclusion criteria are applied to the core projects that are used to generate the dependency data.[1] Although this certainly has some effect, it is unlikely it affects the results substantially as of those 15.439 projects, 14.506 (94%) met the inclusion criteria. Although this means that 6% of the dependencies were lost, the vast majority is still present in the final dataset.

Finally, the data collected in the separate stages are merged, and the data entries of projects with fewer than five valid pull requests are removed. Because software changes over time, meaning that older experiences might not be as relevant as recent experiences, this data is further pre-processed using a *"sliding window"* algorithm using a 90-day time window.[2] Simply put, sliding window means that the algorithm iterated through all the development activities in chronological order (sorted using their closing date), and whenever a new entry was added to the *"window,"* all entries older than 90 days (with respect to the new entry) were removed. Then, the metrics described in the next section were calculated for each pull request, using the data entries in the 90-day time window preceding it.

3.2 Metrics

This study considers two types of variables: developer experience and collaboration variables. Developer experience variables make it possible to evaluate the impact of experience within the software ecosystem and within dependent projects, which is necessary to answer RQ1 and RQ2. Similarly, collaboration variables are calculated using social network analysis, making it possible to answer RQ3. This leaves RQ4, which addresses the impact of the measured variables for first-time contributors, which can be calculated by observing whether a user ever submitted a pull request.

3.2.1 Developer Experience Variables

There are various ways to measure developer experience. A vast number of studies have previously taken the number of pull requests submitted [1], [4], [8]–[10] or the number of code commits [1], [4], [7], [9], [10], [13], [26], [34], not considering the experience acquired in alternative sources, like issue tracking systems or the comment section of issues and pull requests. Notably, developers in open-source software communities commonly wield non-developer roles before making a contribution [19]. This study considers the developer experience factors in Table 2.

[1] This decision was made with the mindset *"collect data now and figure out what to do with it later,"* as data collection is a slow process and waiting with this step until a formal set of criteria was defined would likely only cause delays.

[2] This window size was selected by testing various ones. Although 30 days is a more common time window, this yielded too sparse a dataset for ecosystem and dependency experience.

Name	Description
Pull request merge ratio	The fraction of accepted pull requests.
Pull requests submitted	The total number of submitted pull requests.
Pull request comments	The number of comments placed on pull requests.
Issues submitted	The number of submitted issues.
Issue comments	The number of comments placed on issues.

Table 2: Overview of the considered developer experience types.

Given a software ecosystem, developer experience can be measured at different levels: intra-project, ecosystem-wide, and in projects with an incoming and outgoing dependency. Here, *"ecosystem-wide experience"* is defined as all experience acquired in the ecosystem, excluding intra-project experience (i.e., when calculating ecosystem-wide experience with respect to project X_i, it is the sum over all experience in ecosystem projects X_j minus that of X_i). Distinguishing between intra-project experience and ecosystem-wide experience is important, as intra-project experience has shown to be an important factor of pull request acceptance [1], [4], [8]–[10], and would likely confound the results. It is important to note that this is a fundamentally different notion than the one used by Dey *et al.* [15].

Most modern software is built by re-using the functionality of other projects, typically referred to as implementing packages [22]. An example of this is a web application that is built using the React framework: it re-uses the functionality of the framework and builds upon it wherever necessary. When software does this, it is said to have a dependency on the package, as any change in the package could affect and break the projects that implement it. In this thesis, dependency data is acquired from Katz [40].[1]

Generally, dependencies are perceived in two manners: incoming and outgoing. Here, an incoming dependency refers to a dependency from another project to the focal project. For example, for React, the web application is an incoming dependency. Conversely, an outgoing dependency refers to the inverse: to the web application, React is an outgoing dependency. Although previous work has suggested that dependency networks are not always congruent with the social structure of a community [49], in the cases where they are congruent, it might have some impact.

The work of Dey *et al.* [15] explored the impact of experience acquired in incoming dependencies, measured as a binary variable, suggesting that experience acquired in these projects matters. Answering this research question then becomes important for two reasons: 1) does the direction of the dependency matter? As this has not been studied yet, and 2) to what extent does the level of involvement in those projects matter?

Incoming and outgoing dependency experience is measured very similarly to ecosystem experience, however, with the added requirement that there exists an incoming/outgoing dependency between the two projects. Put concretely, when calculating the incoming dependency experience for project X_i, it is the sum of all experience in ecosystem projects X_j such that there exists a dependency $X_j \rightarrow X_i$. For outgoing dependency experience such that there exists a dependency $X_i \rightarrow X_j$ (i.e., the inverse). As each of the experience types (Table 2) is measured for each level (i.e., intra-project, ecosystem-wide, or in a dependency project), 20 features are measured, making it possible to answer RQ1 and RQ2.

3.2.2 Collaboration Variables and Social Network Analysis

Some studies have already stepped beyond the boundaries of singular projects by exploring the social connections between developers, suggesting that both general connectedness [13], [14], [39] and direct connectedness with your pull request integrator [11], [12] can, but does not ensure [4], a positive impact on pull request acceptance. In these studies, social connectedness is measured using the follower feature in GitHub.

[1]This study included both regular and development dependencies. Katz [40] differentiates between dependencies listed in the repository and those listed in the NPM archive. This study includes both of these to maximise the amount of developer experience observed in the dataset.

Although a follower network provides insights into the social component of software engineering, it can hardly serve as a measurement of professional connectedness. This is because it is accessible to anyone and follow-connections can be made for arbitrary reasons. Therefore, this study includes a new measure of connectedness: connectedness through professional collaboration.

Collaboration in an ecosystem refers to the extent to which a contributor cooperates with others, gaining experience and potentially establishing a sense of trust with other developers. In this study, collaboration is considered in two manners: direct collaboration and indirect collaboration, both of which can be measured using social network analysis. Direct collaboration refers to the number of times two developers have directly cooperated (for example, by discussing something in an issue), emphasising the number of collaborations. Similarly, indirect collaboration is measured to account for the quality of the interaction, as cooperating with well-experienced developers might be more valuable than cooperating with inexperienced developers. This study considers five types of collaboration, which are shown in Table 3.[1]

Social network analysis is an extension to graph theory that explicitly addresses social dynamics within groups of individuals and is commonly applied in software development research to study project organisation and knowledge management [50]–[52]. Although many analysis methods and metrics exist [50], [53], this study applies link strength and node centrality showing how well-connected two nodes are and how important a node is in the graph, respectively.

Using graph theory, a software ecosystem can be defined as a multi-layered graph $\Gamma = (V, L, E)$, where V is a set of nodes (i.e. contributors), L a set of layers (in this study the development activity types shown in Table 3), and $E = \{E_\lambda \mid \lambda \in L\}$ the *set of sets of edges* of the different layers, such that the set E_λ contains all edges of layer type λ, and each edge $\langle a, b, t \rangle \in E_\lambda$ is a triple containing the source and target node a, b as well as a timestamp t at which the edge is created (i.e. when the development activity occurred). In this study, it is equal to the closing time of a pull request or issue. Considering time is important as this makes it possible to consider events chronologically.

[1] These five were selected because they are used in most projects. In GitHub, pull requests can have *"assignees"* and *"reviewers"* too, however, these are less frequently used.

Name	Description	Weight
PR submitted and integrated	Collaboration activity where person x integrates the pull request submitted by person y.	0.247
PR submitted and commented	Collaboration activity where person x comments on the pull request submitted by person y.	0.243
PR discussion participation	Collaboration activity where person x and y both participated in the comment section of a pull request.	0.193
Issue submitted and commented	Collaboration activity where person x comments on the issue submitted by person y.	0.237
Issue discussion participation	Collaboration activity where person x and y both participated in the comment section of an issue.	0.079

Table 3: Overview of the considered collaboration activities and their edge weights.

Using this definition, link strength (i.e., direct collaboration) is the number of edges connecting two nodes, and node centrality (i.e., indirect collaboration) is the general importance of the node in the graph. Various well-known metrics for global centrality exist, like HITS or PageRank [53]. However, their high computation time makes it impossible to use them with this study's granularity level.

Alternative methods are betweenness centrality, which estimates a node's centrality by how often it connects different clusters in a graph, or degree centrality, which estimates a node's centrality by the number of connections it has [53]. As this study attempts to measure indirect collaboration, an alternative of degree centrality is used: first-order degree centrality.[1] This method is suitable for this study as it measures the centrality of a node by counting the collaborations of the nodes it is connected to; i.e., indirect collaboration.

In this study, both in-degree and out-degree are calculated. First-order in-/out-degree centrality can be calculated using the following formulas:

$$\delta^{in}(a,t,\lambda,\mu) = \sum_{\langle b_i,t_i\rangle}^{E_\lambda^{in}(a,t)} d_\mu^{in}(b_i,t_i) - |E_\mu(a,b_i,t_i)| \tag{1}$$

$$\delta^{out}(a,t,\lambda,\mu) = \sum_{\langle b_i,t_i\rangle}^{E_\lambda^{in}(a,t)} d_\mu^{out}(b_i,t_i) - |E_\mu(b_i,a,t_i)| \tag{2}$$

Where $a \in V$, t is the time the centrality is measured, and $\lambda,\mu \in L$. Here, the intuition is that to calculate the first-order in-degree for node a, the μ in-degree of all neighbours b that are connected to a through a λ-edge are summed. Here, the edges that connect a and b_i directly are subtracted as this is equal to the number of times they collaborated directly, defying the point of measuring indirect collaboration. Here, t is included to account for the time at which the metric is calculated and the chronology of the collaborations between a, b_i, and the neighbours of b_i. Concretely, this is used to ensure only collaborations between a and b_i are considered before t and that only collaborations of b_i and its neighbours are considered that happened before t_i, the time at which a and b_i collaborated. This is important, as not doing so would likely inflate the calculated metrics as it unintentionally assumes events in at time t could have affected events at time $t-1$ [54].

Instead of calculating link strength and first-order degree centrality separately for each of the edge types, this study aggregates the results. This decision was made to reduce the complexity of the statistical and predictive analysis described in Section 3.3. To aggregate the metrics of the multiple layers, this study adopts a method described by Kivelä *et al.* [55], which is to simply normalise by taking the weighted sum of the metric calculated per layer using the reciprocal of the total number of edges in that layer (i.e., $w_\lambda = \frac{1}{|E_\lambda|}$)[2]. (Although this is technically true, the implementation does this slightly differently, as the weights are normalised afterwards, for which we have $\sum_\lambda^L w_\lambda = 1$ as otherwise, the resulting values are very small due to a very large number of edges in each layer.)

[1] Although, in hindsight, the study should have included betweenness centrality as well as this could have highlighted the importance of developers that connect different clusters of people (which is what knowledge brokers very likely do). Calculating this using the full graph would likely have been computationally difficult. However, calculating it within a neighbourhood around a node should be completely feasible. It was not included because of simple oversight.

[2] Some testing was done using the analytic hierarchy process [56], however, was ultimately excluded because no notable effect was observed.

The used weights are shown in Table 3. Because first-order degree considers two edge types (the connecting edge λ and the experience edge μ), the weight of both edges is applied:

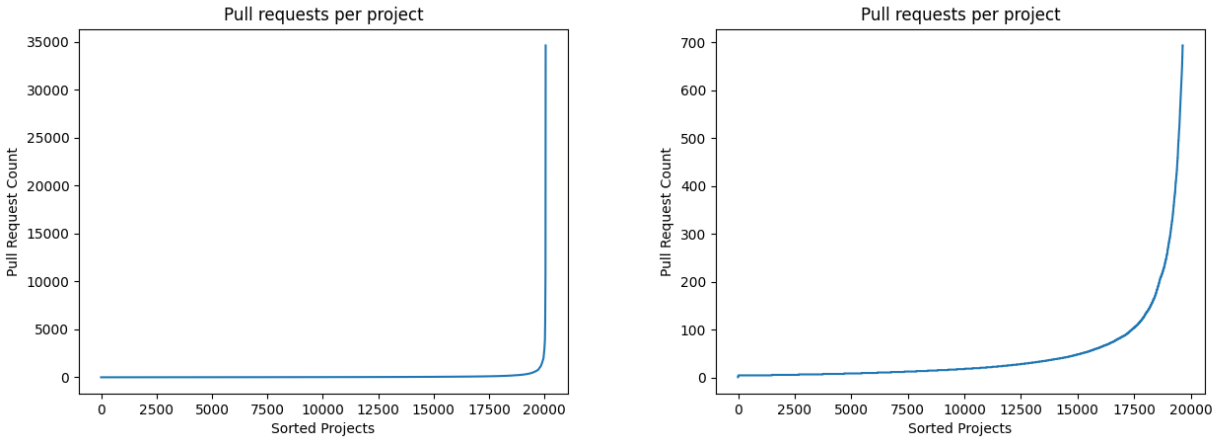
$$C^{in}(a,t) = \sum_{\lambda,\mu}^{L} w_\lambda \cdot w_\mu \cdot \delta^{in}(a,t,\lambda,\mu) \quad (3) \qquad C^{out}(a,t) = \sum_{\lambda,\mu}^{L} w_\lambda \cdot w_\mu \cdot \delta^{out}(a,t,\lambda,\mu) \quad (4)$$

3.3 Data Analysis

This study uses mixed-effects logistic regression to describe the linear impact of the measured variables, as it can calculate general coefficients while respecting random effects across projects. To place the calculated effects in context, control variables for general pull request acceptance research introduced by Zhang *et al.* [4] are used (shown in Table 1). Although Zhang *et al.* suggests including *"whether continuous integration is used"*, it is not included as this data is non-trivial to collect due to the diversity of CI platforms. Similarly, they suggest including *"whether the pull request submitter is a core member,"* which is excluded because it is multicollinear with *"intra-project pull request merge ratio."* Finally, although *"integrator experience"* is included in the random forest models, it is excluded from the regression models as it interacts with the tested variables, causing inconsistent results.

To correctly estimate the effect of the variables, three models are created: ecosystem experience, dependency ecosystem experience, and collaborative experience. This makes it possible to answer RQ1, RQ2, and RQ3. In addition to that, a distinction is made between first-time and non-first-time contributors, making it possible to answer RQ4.

When observing pull request distribution across projects, it showed that the top 2% of the projects were responsible for 49% of the data, which might skew the results. To account for this, a cap of 694 pull requests is put on the number of pull requests from the top 2% projects. This matches the number of pull requests of the largest project that is not part of the top 2%. The pull requests from these projects are randomly sampled.



(a) Pull requests per project.

(b) Pull requests per project up to the 98th percentile.

Figure 2: Distribution of pull requests across projects.

Mixed effects logistic regression models make three assumptions about the used data: linearity between the variables and the log-odds ratio, absence of multicollinearity, and no strong outliers. To

improve the log-linearity of the models, several variables were transformed using add-one log transformation (i.e. $x' = \ln(x+1)$).[1] Multicollinearity is tested using the variance inflation factor (VIF) and using the Spearman correlation coefficient between pairs of features. Variables that strongly correlated with another ($|\rho| \geq 0.5$) or were highly multicollinear ($VIF \geq 5$) were removed.

An exception to this rule is when *"PR has comments"* and *"PR has comment by external contributor,"* as these correlate strongly with each other. After a thorough examination, this interaction was included as it only affected the variables involved and no others. Finally, outliers were removed using Cook's distance, using a cut-off threshold of $4/(n-k-1)$, where n is the number of observations, and k is the number of predictors. This removed between 0.1% and 2.6% of the observations.

Finally, to identify the predictive strength of each feature and feature group, random forest is used [57], [58]. To measure the importance of individual factors, the mean decrease in Gini is calculated, which estimates the amount of information lost when a predictor is removed [57], [58]. To evaluate the predictive strength of each variable, separate models are trained using subsets of features. Their performance is evaluated using the F1 metric (see Equation (7)), which is calculated using 5-fold cross-validation.

$$precision = \frac{TP}{TP+FP} \quad (5) \qquad recall = \frac{TP}{TP+FN} \quad (6) \qquad F1 = 2 \times \frac{prec \times rec}{prec + rec} \quad (7)$$

For context, the mean decrease in Gini (also referred to as *"mean decrease in impurity"*) represents the amount of information lost as indicated by the Gini impurity function after removing a predictor. The Gini function (shown in Equation (8)) is a measure of the amount of information in a set, which random forest models use to evaluate the impact of a decision rule. The Gini function uses the probability of observing a class (in this case merged vs. rejected pull requests) given some decision rule t (e.g., *"does the PR have comments?"*). If after applying this rule, the uniformity of the set increases (i.e., the probability of observing one specific class becomes very high), the amount of information in that set decreases as the rule extracted that amount of information. Given a set of pull requests, random forest models evaluate the impact of a decision rule by comparing the amount of information before applying it versus the amount of information after applying it. If the amount of information in the subsequent sets[2] is lower, the amount of information gain increases. For a single decision tree (as a random forest model is simply an ensemble of many decision trees), the decrease in Gini then represents the amount of information that is lost after a variable is removed from the dataset (i.e. when the decision rules related to that variable are removed). Consequently, the mean decrease in Gini is simply the mean of the decrease in Gini of the separate trees. Most generally, the mean decrease in Gini is a value between 0 and 1, however, technically does not have an upper bound of 1 as it is bound by the number of decision rules related to the variable.

$$Gini(t) = 1 - \sum_{c_i}^{C} p(c_i \mid t)^2 = 1 - \left(p(PR\ merged \mid t)^2 + p(PR\ rejected \mid t)^2 \right) \quad (8)$$

[1] This transformation was chosen after a thorough evaluation of various others and had the overall best impact on the log-linearity of the model. The data distributions of the transformed data can be found in Appendix A and the log-linearity in Appendix B. Appendix C elaborates on the multicollinearity process.

[2] *Sets*, plural, because a decision rule virtually splits a dataset into two parts: one that complies with the rule and one that does not.

	Ecosystem Model	**Dependency Model**	**Collaborative Model**
Control Variables	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*
Pull request is self-integrated	$-0.261^{*}(0.001)$	$-0.255^{*}(0.001)$	*n/a*
Pull request has comments	$-0.127^{*}(0.001)$	$-0.130^{*}(0.001)$	$-0.108^{*}(0.001)$
Pull request contains a "#"	$0.037^{*}(0.001)$	$0.036^{*}(0.001)$	$0.038^{*}(0.001)$
First-time contributor	$-0.129^{*}(0.001)$	$-0.141^{*}(0.001)$	$-0.062^{*}(0.001)$
PR has comment from ext. contr.	$0.031^{*}(0.001)$	$0.030^{*}(0.001)$	$-0.005^{*}(0.001)$
Pull request lifetime$^{\diamond}$	$-0.482^{*}(0.002)$	$-0.484^{*}(0.002)$	$-0.427^{*}(0.002)$
Pull request commit count$^{\diamond}$	$0.191^{*}(0.005)$	$0.170^{*}(0.005)$	$0.064^{*}(0.005)$
Exp. & Collab. Variables	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*
Intra-Pull project issues$^{\diamond}$	$0.272^{*}(0.003)$	$0.286^{*}(0.003)$	*n/a*
Ecosystem Pull requests$^{\diamond}$	$0.252^{*}(0.002)$	*n/a*	*n/a*
In-dependency Pull requests$^{\diamond}$	*n/a*	$0.232^{*}(0.005)$	*n/a*
Out-dependency Pull requests$^{\diamond}$	*n/a*	$0.140^{*}(0.006)$	*n/a*
Weighted first-order in-degree$^{\diamond}$	*n/a*	*n/a*	$0.144^{*}(0.004)$
Integrator to submitter link strength$^{\diamond}$	*n/a*	*n/a*	$0.380^{*}(0.004)$

Table 4: Overview of the coefficients and standard error calculated by the mixed effects logistic regression models, describing the general relationships. $^{\diamond}$Log-transformed variable; $^{*}p < 0.001$.

	Ecosystem Model		**Dependency Model**		**Collaborative Model**	
	FTC	*non-FTC*	*FTC*	*non-FTC*	*FTC*	*non-FTC*
Control Variables	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*
Pull request is self-integrated	$-0.562^{*}(0.002)$	$-0.125^{*}(0.001)$	$-0.563^{*}(0.002)$	$-0.122^{*}(0.001)$	*n/a*	*n/a*
Pull request has comments	$-0.155^{*}(0.002)$	$-0.116^{*}(0.001)$	$-0.161^{*}(0.002)$	$-0.118^{*}(0.001)$	$-0.044^{*}(0.002)$	$-0.106^{*}(0.001)$
Pull request contains a "#"	$0.045^{*}(0.001)$	$0.031^{*}(0.001)$	$0.045^{*}(0.001)$	$0.031^{*}(0.001)$	$0.047^{*}(0.002)$	$0.031^{*}(0.001)$
PR has comment from ext. contr.	$0.024^{*}(0.002)$	$0.048^{*}(0.001)$	$0.023^{*}(0.002)$	$0.048^{*}(0.001)$	$-0.067^{*}(0.002)$	$0.037^{*}(0.001)$
Pull request lifetime$^{\diamond}$	$-0.503^{*}(0.003)$	$-0.493^{*}(0.002)$	$-0.510^{*}(0.003)$	$-0.495^{*}(0.002)$	$-0.205^{*}(0.003)$	$-0.465^{*}(0.002)$
Pull request commit count$^{\diamond}$	$0.216^{*}(0.010)$	$0.285^{*}(0.005)$	$0.192^{*}(0.010)$	$0.275^{*}(0.005)$	$-0.149^{*}(0.012)$	$0.223^{*}(0.005)$
Exp. & Collab. Variables	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*	*Coef. (std. err.)*
Intra-project issues$^{\diamond}$	$0.259^{*}(0.016)$	$0.170^{*}(0.003)$	$0.284^{*}(0.016)$	$0.179^{*}(0.003)$	*n/a*	*n/a*
Ecosystem Pull requests$^{\diamond}$	$0.385^{*}(0.004)$	$0.164^{*}(0.002)$	*n/a*	*n/a*	*n/a*	*n/a*
In-dependency Pull requests$^{\diamond}$	*n/a*	*n/a*	$0.606^{*}(0.018)$	$0.151^{*}(0.004)$	*n/a*	*n/a*
Out-dependency Pull requests$^{\diamond}$	*n/a*	*n/a*	$0.500^{*}(0.016)$	$0.092^{*}(0.006)$	*n/a*	*n/a*
Weighted first-order in-degree$^{\diamond}$	*n/a*	*n/a*	*n/a*	*n/a*	$0.351^{*}(0.010)$	$0.083^{*}(0.003)$
Integrator to submitter link strength$^{\diamond}$	*n/a*	*n/a*	*n/a*	*n/a*	$0.595^{*}(0.016)$	$0.350^{*}(0.003)$

Table 5: Overview of the coefficients and standard error calculated by the mixed effects logistic regression models, distinguishing between first-time contributors and non-first-time contributors. $^{\diamond}$Log-transformed variable; $^{*}p < 0.001$.

4 Results

The following section describes the results of the analysis. Section 4.1 provides a general overview of the results, elaborating some descriptive details of the collected dataset, discussing the types of results generated in this study, and addressing the effect of the observed control variables. Then, Section 4.2 addresses each research question, answering them using the presented results.

4.1 General Overview

The collected dataset consists of 1.8 million pull requests and 2.1 million issues spread across 20.052 projects, submitted by 190.898 unique users, across a period of 9 years (visualised in Figure 3). Of these, 72% of the projects were collected using the popularity criterion and 28% with the dependency criterion, which is almost equally split into projects with an incoming and an outgoing dependency on a popular project.

In the complete dataset, 17.280 projects have an outgoing dependency on another project, whereas only 2.217 have an incoming dependency, suggesting a core-periphery structure in the dependency network. Since few projects are responsible for the majority of the pull requests, for representativeness, only 1.2 million data points are used for statistical and predictive modelling.

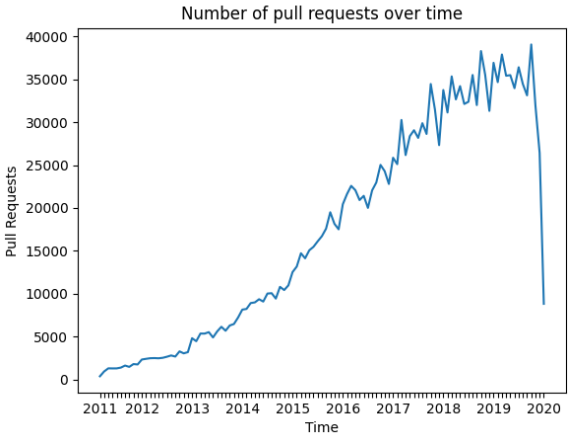


Figure 3: Distribution of pull requests across time. The stark decline in 2020 is because only data is collected up to early January 2020.

Table 5 presents the results of the mixed effects logistic regression models. The calculated coefficients suggest the direction of the effect of the predictor, such that negative values decrease the odds of acceptance and positive values increase them. During the analysis phase, it became apparent that all experience types strongly positively correlate with each other. Therefore, only one developer experience variable is included in the model per experience level (i.e., intra-project, ecosystem-wide, etc.). Concretely, this means that ecosystem-wide, incoming and outgoing dependency and intra-project experience, most experience variables shown in Table 2 had a Spearman coefficient greater than 0.5. Similarly, collaboration metrics calculated using either in- or out-degree strongly correlated as well.[1]

[1] For a complete overview of correlating factors, refer to Appendix C.

The results show that each measured variable significantly relates to pull request acceptance. Here, the effect of each control variable is largely in line with the results presented by Zhang *et al.* [4], having only one difference: the impact of a comment placed by an external contributor, which became negative. However, this effect is likely due to an interaction with *"PR is self-integrated"* as the same is observed when that variable is removed from other models. Interestingly, however, the non-first-time contributor model does align with prior research.

To identify the importance of the used features, Figure 4 provides an overview of the mean decrease in Gini of the top ten most important features of the random forest model trained on the full dataset as well as their respective scores of the models trained using the (non-)first-time contributor subsets. Note that the FTC model has such low scores for intra-project pull request experience because they do not contain any information by definition of *"first-time contributor"* as they have not submitted a single pull request yet.

It stands out that a relatively large amount of information is acquired from the control variables: *PR lifetime, integrator experience, self-integrated PRs,* and *commit count*. This is not surprising as this is what these variables are included for. Conversely, the absence of *comment count* and *comment of external contributor* is interesting as this suggests various newly added features wield more important information than these two control variables.

Table 6 provides the broadest view of the analysis, showing the F1 scores calculated using random forest models trained using subsets of predictors. When observing the ratio of merged pull requests, it became apparent that there exists a considerable class imbalance in each of the used data models, as there are many more merged pull requests than rejected ones. To account for this, Table 6 includes a baseline for each model equal to the probability of observing a merged pull request in that dataset. Therefore, although any model with an F1 score over 0.5 would have some predictive power, only models outperforming this baseline perform better than a probabilistic guesser.

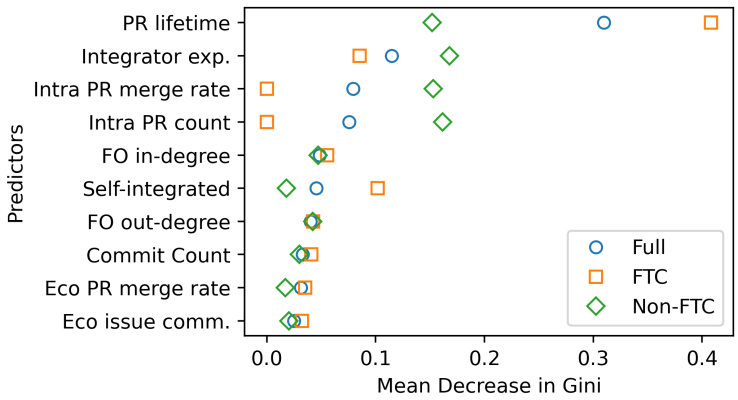


Figure 4: Feature importance plot showing the mean decrease in Gini calculated using random forest trained using the full dataset and the (non-)first-time contributor subsets.

4.2 Answering Questions

4.2.1 Ecosystem-wide Experience

The analysis phase showed that the measured experience types (i.e., pull requests submitted, comments on issues, etc.) positively correlate, for which only one is included in the regression models. The results of the logistic regression models show that ecosystem-wide experience positively impacts pull request acceptance, even when accounting for intra-project experience. This suggests that the findings of Dey *et al.* [15] generally hold. However, a notable difference is that they included both pull request submission count and pull request merge rate, measuring a negative effect of submission count. The results of this study initially showed this effect with each experience type too (excluding merge ratio). However, it disappeared when testing these variables separately. This suggests the effect was likely due to an unaccounted-for interaction between variables. When comparing the importance of ecosystem-wide experience with intra-project experience, it shows that their predictive power and coefficients are very similar. However, removing intra-project experience does yield a higher loss of information in the random forest models.

RQ1: ***Does prior experience (including non-code contributions) within the ecosystem influence the chances of developer pull request acceptance in another project within the ecosystem?***

Yes. The experience acquired through submitting pull requests, submitting issues, and participating in their discussions has a significant positive effect on pull request acceptance in another project in the same ecosystem. Compared to intra-project experience, the effect size and predictive power of ecosystem-wide experience are very similar. However, intra-project experience did result in a higher loss of information when removed from a random forest model.

4.2.2 Experience in Dependent Projects

Although it is important to understand the general impact of ecosystem experience, what makes ecosystems special is that projects can have technical dependencies on one another. Consequently,

Predictors	Full Model	FTC Model	nFTC Model
Baseline	0.788	0.659	0.857
Control	0.885 (0.0003)	0.797 (0.0008)	0.921 (0.0005)
Intra-project	0.881 (0.0007)	0.794 (0.0010)	0.925 (0.0004)
Combined	0.911 (0.0006)	0.809 (0.0008)	0.957 (0.0004)
Ecosystem	0.877 (0.0007)	0.783 (0.0012)	0.920 (0.0003)
In-dependency	0.881 (0.0004)	0.794 (0.0013)	0.923 (0.0004)
Out-dependency	0.881 (0.0005)	0.794 (0.0008)	0.923 (0.0005)
Collaboration	0.872 (0.0005)	0.769 (0.0011)	0.917 (0.0006)
Combined	0.874 (0.0004)	0.777 (0.0009)	0.920 (0.0004)
Full Model	0.923 (0.0004)	0.833 (0.0007)	0.961 (0.0003)

Table 6: The mean and standard deviation of the F1 scores calculated with the random forest models, calculated using 5-fold cross-validation. The different models are trained using the full dataset and the (non-)first-time contributor datasets. The baseline is equal to the probability of observing a merged pull request.

these projects have some explicit technical relevance towards one another as experience with implementing a package could help build it further (an in-dependency). Similarly, experience in building a package could help to implement it (an out-dependency). When observing the results in Table 5 it stands clear that in- and out-dependency experience is relevant as both are positively related to pull request acceptance. However, the effect of experience in out-dependent projects is slightly lower.

These findings align with the prior work by Dey *et al.* [15] and add that both in- and out-dependencies are relevant, not just in-dependencies. Regardless of its relatively large coefficients, it stands out that the importance scores in the random forest model for dependency experience are relatively low (none of them appears in the top ten of any model), suggesting that they have relatively little predictive value in general. This could, however, be explained by the rarity of dependency experience as only $\pm 10\%$ of all data points have non-zero values for dependency experience. This suggests that although its impact is relatively high, no information can be derived from it for most observations.

RQ2: ***Does prior experience in projects involved in a technical dependency influence developer pull request acceptance in the projects they have a technical dependency with?***

Yes. The experience acquired in in- and out-dependent projects has a significant positive effect on pull request acceptance. Of these, experience acquired in in-dependent projects is relatively more important. However, its overall predictive strength is lower because relatively few developers have experience in dependent projects.

4.2.3 Direct and Indirect Collaboration

This study addressed collaboration in software ecosystems in twofold: direct and indirect collaborations. Both of these measures were calculated using a directed graph. However, the data analysis showed that direction does not matter as they positively correlate. The results in Table 5 showed that each of these measures has a significant positive effect on pull request acceptance. These results suggest that the number of direct collaborations with pull request integrators and collaboration with experienced developers are important indicators of a developer's success within a software ecosystem. The results in Table 5 suggest that direct collaboration is of greater importance than indirect collaboration. This could be explained by it being more tangible. However, figure 4 shows that direct collaboration provides relatively little additional information compared to indirect collaboration. Compared to previous work, then, this suggests that both the social connectedness [4], [11]–[14], [39] and professional connectedness have a beneficial impact on a developer's success.

RQ3: ***Does collaboration within the ecosystem influence the chances of developer pull request acceptance in another project within the ecosystem?***

Yes. Both direct collaborations with pull request integrators and indirect collaboration have been shown to increase pull request acceptance in software ecosystems. This suggests that both the number of collaborations and their quality matter when participating in an open-source software ecosystem. Although the logistic regression models suggest that direct collaboration has a greater effect on pull request acceptance, indirect collaboration has been shown to provide more information to the random forest model.

4.2.4 First-time Contributors

Prior research has shown that unit changes proposed by first-time contributors are more likely to be rejected [7], [26], [29], [31], a finding that is successfully replicated in this study as well. Conversely, the theory proposed by Rastogi *et al.* [21] suggested that first-time contributors with experience in the same ecosystem have greater odds of their proposed changes being accepted with the expectation that an increased trust is established. When observing the results presented in Table 5 it is clear that their expectation was right as the impact of all of the measured ecosystem-wide experience and collaborative factors are positive and substantially higher for first-time contributors compared to non-first-time contributors.

Notably, intra-project experience is more relevant in this scenario as well. This could suggest that active participation within the project before submitting a pull request benefits first-time contributors. When comparing the coefficients of ecosystem-wide experience to intra-project experience, however, it shows that their impact is substantially higher. The combination of all these factors shows the impact of: 1) prior domain knowledge and 2) a connection built with the development team before proposing a unit change. In the case of intra-project experience, a potential explanation is that participation in discussion before submitting pull requests could shed light on details the contributors might have otherwise overlooked. Similarly, it is possible that participating in these discussions might allow them to prove their capabilities to the development team, increasing their trust. These findings are in line with prior work on first-time contributors who identified a lack of technical expertise as a barrier [20], [32], [33]. This study, then, suggests that prior domain experience lifts this barrier.

RQ4: ***Does prior experience within the ecosystem influence the chances of first-time contributor pull request acceptance in another project within the ecosystem?***

Yes. The results presented in this study show a substantial difference between the impact of intra-project experience, (dependency) ecosystem-wide experience, and direct and indirect collaboration for first-time contributors when compared to non-first-time contributors. The study shows that prior participation inside a software ecosystem by submitting pull requests and issues or participating in their discussion benefits newcomers. The results of ecosystem-wide experience suggest that understanding the system positively benefits a first-time contributor's odds of pull request acceptance. The same applies to prior collaboration with pull request integrators.

5 Discussion

The following section briefly discusses the presented results. Section 5.1 describes some implications of this work concerning developers and researchers. Then, Section 5.2 describes the threats to the validity of this work.

5.1 Implications

5.1.1 Implications for (First-Time) Contributors

The results of this study suggest that experience acquired through posting issues, submitting pull requests, and participating in their discussions is valuable. This holds for experience within a project, in a project with an incoming or outgoing dependency, or within the same ecosystem This is true for all developers and especially for first-time contributors. Furthermore, the results suggest that collaborating with experienced developers and your pull request integrator is also beneficial. This suggests that who you collaborate with should be an active consideration, not just whether you collaborate.

5.1.2 Implications for Software Teams

The results show that pull requests submitted by contributors with ecosystem experience are more likely to be accepted and merged. A similar effect is measured with developers who are professionally closely connected with project members. This implies that the success of newcomer onboarding could be increased when considering their prior experience in the ecosystem. This could go as far as recruiting potential developers who contributed to a dependent project or a developer a core member priorly collaborated with.

5.1.3 Implications for Researchers

This study expanded the knowledge on pull-based development in open-source software ecosystems by including experience acquired in issue-tracking systems at an intra-project, project dependency, and ecosystem level. This study empirically confirms the hypothesis presented by Rastogi *et al.* [21] to show that the ecosystem-wide experience could influence the success of first-time contributors in a project within the software ecosystem.

Further, the study adds that applying social network analysis to identify direct and indirect connections between developers can indicate the knowledge flow and inner social structures in open-source ecosystems. The results suggest that ecosystem-wide factors improve predicting the outcome of pull requests submitted by first-time contributors. However, classifying these remains difficult, suggesting a direction for future research.

5.2 Threats to Validity

5.2.1 Internal Validity

Since the core collection of projects used in this study was generated based on their popularity, this collection might misrepresent the NPM ecosystem. Although a set of periphery projects was added to account for technical dependencies among projects, which included various less popular projects, this does not ensure that the general representation of NPM projects improved.

Another consequence of this method is a stark skewness in the distribution of pull requests across projects such that only 2% of the projects contained 49% of the pull requests. To account for this, the pull requests of these dominating projects have been capped and randomly sampled to limit their effect.

Previous work identified that people commonly use multiple aliases in open-source software ecosystems [59] and that accounting for these can substantially affect the structure of social networks [60]. Although various solutions exist in the literature [60], [61], none of these is applied here due to stringent data requirements and the substantial amount of manual labour required to apply these methods at an ecosystem scale accurately.[1]

The dataset created by Katz [40] is the foundation of this study because of the large amount of archival data it contains. A consequence of this is that our study inherits the flaws in this dataset, one of which is the potential accuracy of the identified dependencies. The dependencies measured in the dataset are generated using the data contained in the NPM and GitHub. The data, however, represent a snapshot in time with no explicit regard for the changes that happen over time. Therefore, although the listed dependencies were likely correct at some point, they might not accurately represent the project's dependencies across time. Similarly, some dependencies might have been missed because various projects do not explicitly list them.

5.2.2 Construct Validity

Many methods exist to identify the relationship between independent factors and a binary independent variable. This study employs two of them: mixed effects logistic regression to identify the general relationship between features and random forest to identify the predictive power of each feature and each feature group. To ensure that the measured effects reflect reality, several control variables suggested by Zhang *et al.* [4] that are known to affect pull request acceptance significantly have also been included in this study. Furthermore, although this study intended only to include development activities performed by humans by filtering bots using the lists composed in previous research [41], [44]–[46] as well as those identified in a manual sweep of submitters with over 400 pull requests, it cannot be assured that no bots are included in this study.

5.2.3 External Validity

This work exclusively addresses the tested hypothesis in the NPM ecosystem. There exist many open-source software ecosystems, each serving different purposes (e.g., packaging ecosystems like NPM compared to functional ones like OpenStack). The conclusions drawn here might not represent these other ecosystems and should be verified in future research.

[1] For example, conservatively applying the method proposed by Vasilescu *et al.* [61] required that the 45K identified matches were checked manually because it had an estimated error rate of $\pm 60\%$. Their method is still a substantial improvement over manually identifying these. However, it is inapplicable in ecosystem-scale research where time is limited and alias unmasking is not a primary goal.

6 Conclusion

The goal of this thesis was to identify the influence of ecosystem-wide experience and collaboration on pull request acceptance in open-source software ecosystems. Understanding this relationship is important for two reasons. Firstly, modern software projects rarely exist in complete isolation because they commonly depend on other projects in the same ecosystem. Therefore, understanding the dynamics of software ecosystems is paramount to understanding software development as a whole. Secondly, developers and projects seek meaningful ways to engage in open-source software development. Therefore, understanding how they can do this meaningfully is important to their success.

This study explored the effect of four topics: 1) ecosystem-wide experience, 2) experience in projects with a dependency, 3) collaboration in an ecosystem, and 4) the impact of ecosystem-wide experience and collaboration on first-time contributors. To test the effect of these factors, a collection of approximately 1.8 million and 2.1 million issues was collected from 20.052 projects sampled from the NPM ecosystem.

The results show that ecosystem-wide experience and collaboration have a significant and positive effect on pull request acceptance. This vouches for the relevance of ecosystem-wide experience and collaboration. However, its relevance is still lower than the experience gained within the project itself. This effect is especially important for first-time contributors as the measured effect sizes are substantially higher for this subgroup. For them, the measured effect size even exceeded the importance of intra-project experience acquired by submitting issues and commenting on pull requests and issues.

The results show that ecosystem-wide and collaboration factors complement previously known intra-project factors of pull request acceptance. The results of the random forest models suggest that predicting the outcome of pull requests submitted by first-time contributors is relatively difficult, yielding only an F1 score of 0.82. However, similarly, pull requests submitted by non-first-time contributors reached an astounding F1 score of 0.96, for which the overall F1 score reached a good 0.92.

6.1 Future Work

There are several directions future work can pursue. First and foremost, this study and the study performed by Dey *et al.* [15] are performed using data acquired from the NPM ecosystem, limiting the generalizability of the combined results. A logical first step for future studies would be to perform a replication study in other package management systems, like PyPI, Maven or Go. Such a study could be set up very similarly to this one because the dataset created by Katz [40] contains information on 32 package repositories. Such a study should not limit itself to package ecosystems, as studying solution-oriented ecosystems like OpenStack or Apache is equally interesting.

This study laid the groundwork for the application of social network analysis in software development research by using a social network built using collaborations. This study suggested the importance of connectedness in such a network, however, did not address this at a deeper level yet. A first step could be to emphasise the impact of knowledge brokers in open-source ecosystems. This could start as simple as calculating the betweenness centrality (an indicator of whether a person connects two otherwise unconnected graph components, and could go as far as to study the socio-technical congruence between projects. To the best of my knowledge, only one case study has been done in this

direction [49] using the Ruby ecosystem, therefore awaiting validation in a larger study. The results of this thesis suggest that in cases where socio-technical congruence exists, it is valuable, for which future work could study how this can be further utilised.

As of now, although a lot is known about open-source software ecosystems and the software within them, a lot is left to be studied still [21]. This thesis attempted to do its small part and hopes to see what follows in the future.

Bibliography

[1] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014, New York, NY, USA: Association for Computing Machinery, May 2014, pp. 345–355, ISBN: 978-1-4503-2756-5.

[2] G. Gousios, A. Zaidman, M.-A. Storey, and A. v. Deursen, "Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, ISSN: 1558-1225, vol. 1, May 2015, pp. 358–368.

[3] S. Panichella, G. Bavota, M. D. Penta, G. Canfora, and G. Antoniol, "How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes," in *2014 IEEE International Conference on Software Maintenance and Evolution*, ISSN: 1063-6773, Sep. 2014, pp. 251–260.

[4] X. Zhang, Y. Yu, G. Georgios, and A. Rastogi, "Pull Request Decisions Explained: An Empirical Overview," *IEEE Transactions on Software Engineering*, pp. 1–1, 2022, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520.

[5] X. Zhang, Y. Yu, T. Wang, A. Rastogi, and H. Wang, "Pull request latency explained: An empirical overview," en, *Empirical Software Engineering*, vol. 27, no. 6, p. 126, Nov. 2022, ISSN: 1382-3256, 1573-7616.

[6] M. Golzadeh, A. Decan, and T. Mens, "On the effect of discussions on pull request decisions," en, 2019.

[7] D. M. Soares, M. L. De Lima Junior, L. Murta, and A. Plastino, "Rejection Factors of Pull Requests Filed by Core Team Developers in Software Projects with High Acceptance Rates," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2015, pp. 960–965.

[8] A. Rastogi, N. Nagappan, G. Gousios, and A. van der Hoek, "Relationship between geographical location and evaluation of developer contributions in github," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '18, New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 1–8, ISBN: 978-1-4503-5823-1.

[9] N. Khadke, M. H. Teh, and M. Shen, "Predicting Acceptance of GitHub Pull Requests," en, 2012.

[10] F. Zampetti, G. Bavota, G. Canfora, and M. D. Penta, "A Study on the Interplay between Pull Request Review and Continuous Integration Builds," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, ISSN: 1534-5351, Feb. 2019, pp. 38–48.

[11] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings of the 36th International Conference on Software En-*

gineering, ser. ICSE 2014, New York, NY, USA: Association for Computing Machinery, May 2014, pp. 356–366, ISBN: 978-1-4503-2756-5.

[12] R. N. Iyer, S. A. Yun, M. Nagappan, and J. Hoey, "Effects of Personality Traits on Pull Request Acceptance," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2632–2643, Nov. 2021, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520.

[13] Y. Yu, G. Yin, T. Wang, C. Yang, and H. Wang, "Determinants of pull-based development in the context of continuous integration," en, *Science China Information Sciences*, vol. 59, no. 8, p. 080 104, Jul. 2016, ISSN: 1869-1919.

[14] M. Soto, Z. Coker, and C. Le Goues, "Analyzing the Impact of Social Attributes on Commit Integration Success," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, May 2017, pp. 483–486.

[15] T. Dey and A. Mockus, "Effect of Technical and Social Factors on Pull Request Quality for the NPM Ecosystem," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ser. ESEM '20, New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 1–11, ISBN: 978-1-4503-7580-1.

[16] O. Franco-Bedoya, D. Ameller, D. Costal, and X. Franch, "Open source software ecosystems: A Systematic mapping," en, *Information and Software Technology*, vol. 91, pp. 160–185, Nov. 2017, ISSN: 0950-5849.

[17] G. K. Hanssen and T. Dybå, "Theoretical foundations of software ecosystems," en, 2012.

[18] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in Empirical Software Engineering," en, *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, Apr. 2008, ISSN: 1573-7616.

[19] J. L. Cánovas Izquierdo and J. Cabot, "On the analysis of non-coding roles in open source development," en, *Empirical Software Engineering*, vol. 27, no. 1, p. 18, Nov. 2021, ISSN: 1573-7616.

[20] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW '15, New York, NY, USA: Association for Computing Machinery, Feb. 2015, pp. 1379–1392, ISBN: 978-1-4503-2922-4.

[21] A. Rastogi and G. Gousios, *How does Software Change?* arXiv:2106.01885 [cs], Jun. 2021.

[22] A. Decan, T. Mens, and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems," en, *Empirical Software Engineering*, vol. 24, no. 1, pp. 381–416, Feb. 2019, ISSN: 1573-7616.

[23] R. Méndez-Durón and C. E. García, "Returns from social capital in open source software networks," en, *Journal of Evolutionary Economics*, vol. 19, no. 2, pp. 277–295, Apr. 2009, ISSN: 1432-1386.

[24] C. Fershtman and N. Gandal, "Direct and indirect knowledge spillovers: The "social network" of open-source projects," en, *The RAND Journal of Economics*, vol. 42, no. 1, pp. 70–91, 2011, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1756-2171.2010.00126.x, ISSN: 1756-2171.

[25] G. Peng, Y. Wan, and P. Woodlock, "Network ties and the success of open source software development," en, *The Journal of Strategic Information Systems*, vol. 22, no. 4, pp. 269–281, Dec. 2013, ISSN: 0963-8687.

[26] D. M. Soares, M. L. de Lima Júnior, L. Murta, and A. Plastino, "Acceptance factors of pull requests in open-source projects," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15, New York, NY, USA: Association for Computing Machinery, Apr. 2015, pp. 1541–1546, ISBN: 978-1-4503-3196-8.

[27] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "The Secret Life of Patches: A Firefox Case Study," in *2012 19th Working Conference on Reverse Engineering*, ISSN: 2375-5369, Oct. 2012, pp. 447–455.

[28] G. Pinto, L. F. Dias, and I. Steinmacher, "Who gets a patch accepted first? comparing the contributions of employees and volunteers," in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '18, New York, NY, USA: Association for Computing Machinery, May 2018, pp. 110–113, ISBN: 978-1-4503-5725-8.

[29] A. Lee and J. C. Carver, "Are One-Time Contributors Different? A Comparison to Core and Periphery Developers in FLOSS Repositories," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Nov. 2017, pp. 1–10.

[30] A. Forte and C. Lampe, "Defining, Understanding, and Supporting Open Collaboration: Lessons From the Literature," en, *American Behavioral Scientist*, vol. 57, no. 5, pp. 535–547, May 2013, Publisher: SAGE Publications Inc, ISSN: 0002-7642.

[31] V. Kovalenko and A. Bacchelli, "Code review for newcomers: Is it different?" In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '18, New York, NY, USA: Association for Computing Machinery, May 2018, pp. 29–32, ISBN: 978-1-4503-5725-8.

[32] D. Wermke, N. Wöhler, J. H. Klemmer, M. Fourné, Y. Acar, and S. Fahl, "Committed to Trust: A Qualitative Study on Security & Trust in Open Source Software Projects," in *2022 IEEE Symposium on Security and Privacy (SP)*, ISSN: 2375-1207, May 2022, pp. 1880–1896.

[33] A. Rastogi, S. Thummalapenta, T. Zimmermann, N. Nagappan, and J. Czerwonka, "Ramp-Up Journey of New Hires: Tug of War of Aids and Impediments," in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ISSN: 1949-3789, Oct. 2015, pp. 1–10.

[34] O. Kononenko, T. Rose, O. Baysal, M. Godfrey, D. Theisen, and B. de Water, "Studying pull request merges: A case study of shopify's active merchant," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP

'18, New York, NY, USA: Association for Computing Machinery, May 2018, pp. 124–133, ISBN: 978-1-4503-5659-6.

[35] D. Legay, A. Decan, and T. Mens, "On the impact of pull request decisions on future contributions," en, in *Belgium-Netherlands Software Evolution Workshop (BENEVOL)*, May 2019.

[36] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "Investigating technical and non-technical factors influencing modern code review," en, *Empirical Software Engineering*, vol. 21, no. 3, pp. 932–959, Jun. 2016, ISSN: 1573-7616.

[37] A. Rastogi, "Do Biases Related to Geographical Location Influence Work-Related Decisions in GitHub?" In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, May 2016, pp. 665–667.

[38] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, and J. Stallings, "Gender differences and bias in open source: Pull request acceptance of women versus men," en, *PeerJ Computer Science*, vol. 3, e111, May 2017, ISSN: 2376-5992.

[39] D. Celińska, "Coding together in a social network: Collaboration among GitHub users," in *Proceedings of the 9th International Conference on Social Media and Society*, ser. SMSociety '18, New York, NY, USA: Association for Computing Machinery, Jul. 2018, pp. 31–40, ISBN: 978-1-4503-6334-1.

[40] J. Katz, *Libraries.io Open Source Repository and Dependency Metadata*, Jan. 2020.

[41] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filippova, and A. Mockus, "Detecting and Characterizing Bots that Commit Code," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20, New York, NY, USA: Association for Computing Machinery, Sep. 2020, pp. 209–219, ISBN: 978-1-4503-7517-7.

[42] M. Golzadeh, D. Legay, A. Decan, and T. Mens, "Bot or not? Detecting bots in GitHub pull request activity based on comment similarity," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20, New York, NY, USA: Association for Computing Machinery, Sep. 2020, pp. 31–35, ISBN: 978-1-4503-7963-2.

[43] N. Chidambaram and P. R. Mazrae, "Bot detection in GitHub repositories," in *Proceedings of the 19th International Conference on Mining Software Repositories*, ser. MSR '22, New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 726–728, ISBN: 978-1-4503-9303-4.

[44] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments," en, *Journal of Systems and Software*, vol. 175, p. 110 911, May 2021, ISSN: 0164-1212.

[45] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filippova, and A. Mockus, *A dataset of Bot Commits*, Jan. 2020.

[46] M. Golzadeh, A. Decan, D. Legay, and T. Mens, *A ground-truth dataset to identify bots in GitHub*, Jan. 2020.

[47] J. Katz, *Libraries.io Open Source Repository and Dependency Metadata*, version 1.6.0, Zenodo, Jan. 2020.

[48] S. Dueñas, V. Cosentino, J. M. Gonzalez-Barahona, A. d. C. S. Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, and A. P. García-Plaza, "GrimoireLab: A toolset for software development analytics," en, *PeerJ Computer Science*, vol. 7, e601, Jul. 2021, Publisher: PeerJ Inc., ISSN: 2376-5992.

[49] M. M. M. Syeed, K. M. Hansen, I. Hammouda, and K. Manikas, "Socio-Technical Congruence in the Ruby Ecosystem," in *Proceedings of The International Symposium on Open Collaboration*, ser. OpenSym '14, New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 1–9, ISBN: 978-1-4503-3016-9.

[50] R. R. Schreiber and M. P. Zylka, "Social Network Analysis in Software Development Projects: A Systematic Literature Review," *International Journal of Software Engineering and Knowledge Engineering*, vol. 30, no. 03, pp. 321–362, Mar. 2020, Publisher: World Scientific Publishing Co., ISSN: 0218-1940.

[51] K. McClean, D. Greer, and A. Jurek-Loughrey, "Social network analysis of open source software: A review and categorisation," en, *Information and Software Technology*, vol. 130, p. 106 442, Feb. 2021, ISSN: 0950-5849.

[52] S. Herbold, A. Amirfallah, F. Trautsch, and J. Grabowski, "A systematic mapping study of developer social network research," en, *Journal of Systems and Software*, vol. 171, p. 110 802, Jan. 2021, ISSN: 0164-1212.

[53] M. Newman, "Measures and metrics," in *Networks*, M. Newman, Ed., Oxford University Press, Jul. 2018, p. 0, ISBN: 978-0-19-880509-0.

[54] P. Holme and J. Saramäki, "A Map of Approaches to Temporal Networks," en, in *Temporal Network Theory*, ser. Computational Social Sciences, P. Holme and J. Saramäki, Eds., Cham: Springer International Publishing, 2019, pp. 1–24, ISBN: 978-3-030-23495-9.

[55] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, Sep. 2014, ISSN: 2051-1310.

[56] V. Casola, A. Fasolino, N. Mazzocca, and P. Tramontana, "An AHP-Based Framework for Quality and Security Evaluation," in *2009 International Conference on Computational Science and Engineering*, vol. 3, Aug. 2009, pp. 405–411.

[57] L. Breiman, "Random Forests," en, *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 1573-0565.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[59] I. S. Wiese, J. T. Da Silva, I. Steinmacher, C. Treude, and M. A. Gerosa, "Who is Who in the Mailing List? Comparing Six Disambiguation Heuristics to Identify Multiple Addresses of a

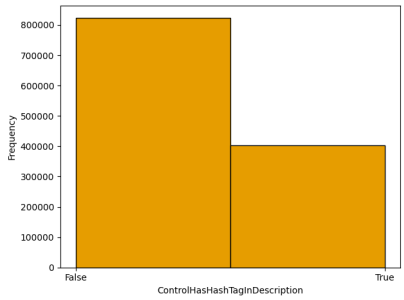
Participant," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Oct. 2016, pp. 345–355.

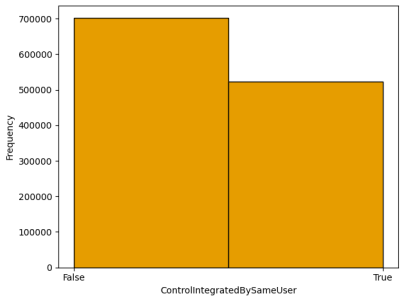
[60] S. Amreen, A. Mockus, R. Zaretzki, C. Bogart, and Y. Zhang, "ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems," en, *Empirical Software Engineering*, vol. 25, no. 2, pp. 1136–1167, Mar. 2020, ISSN: 1573-7616.

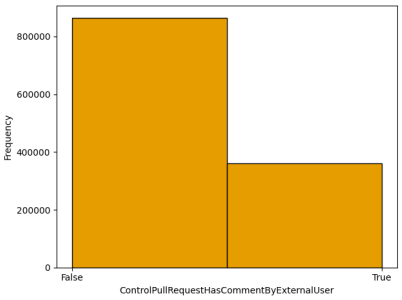
[61] B. Vasilescu, A. Serebrenik, and V. Filkov, "A Data Set for Social Diversity Studies of GitHub Teams," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, ISSN: 2160-1860, May 2015, pp. 514–517.

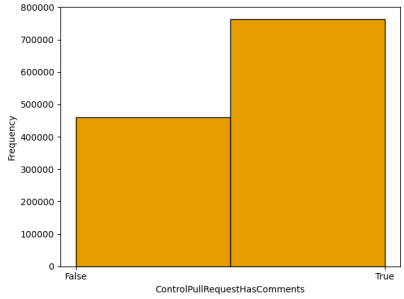
Appendices

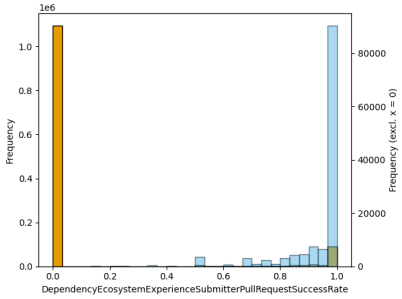
A Data Distributions

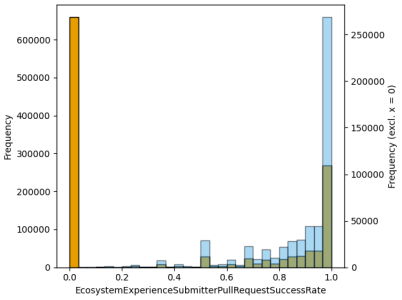


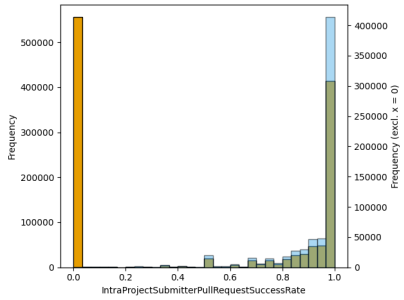


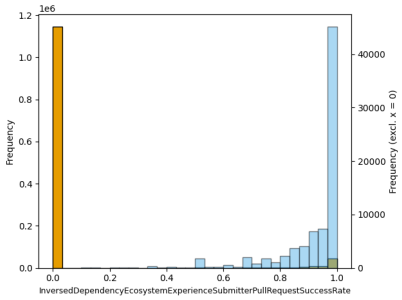


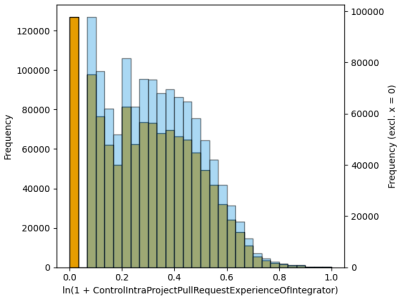


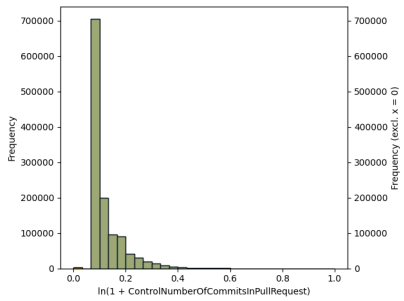


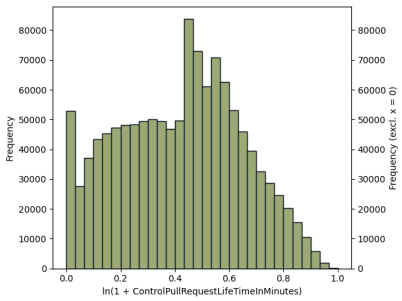


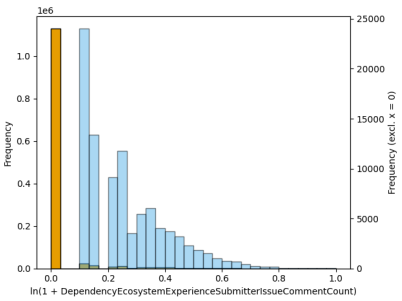


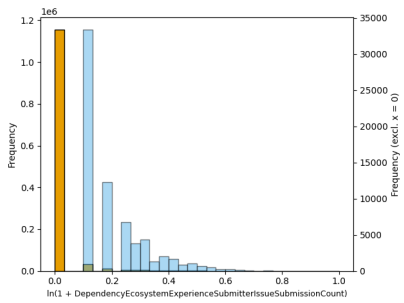


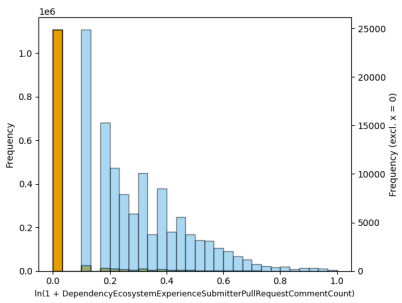


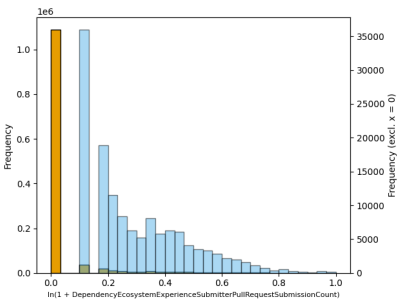


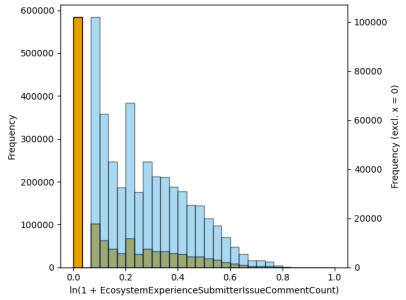


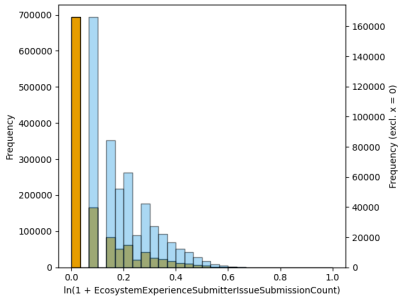


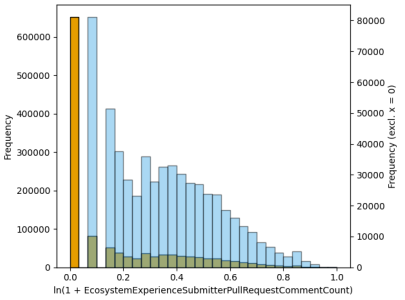


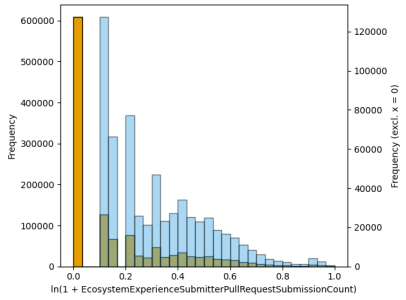


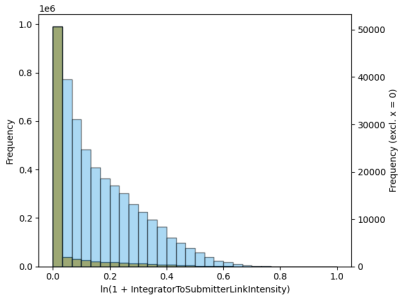


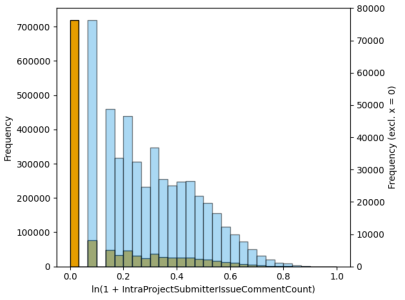

Frequency
Frequency (excl. x = 0)
ln(1 + EcosystemExperienceSubmitterIssueCommentCount)

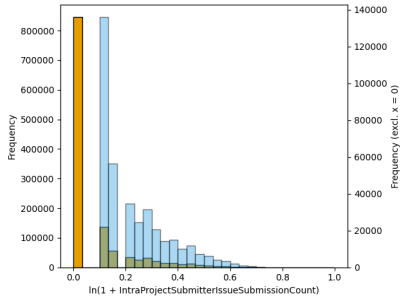

Frequency
Frequency (excl. x = 0)
ln(1 + EcosystemExperienceSubmitterIssueSubmissionCount)

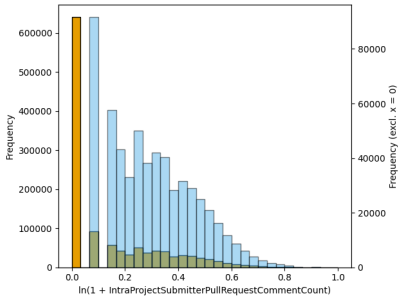

Frequency
Frequency (excl. x = 0)
ln(1 + EcosystemExperienceSubmitterPullRequestCommentCount)

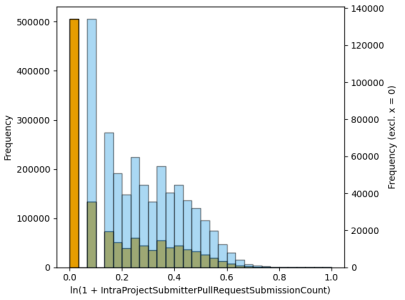

Frequency
Frequency (excl. x = 0)
ln(1 + EcosystemExperienceSubmitterPullRequestSubmissionCount)

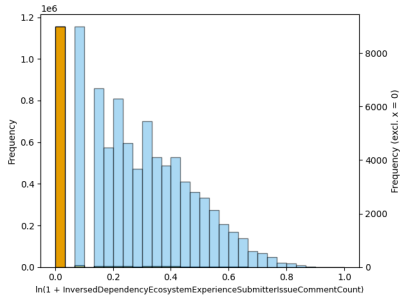

1e6
Frequency
Frequency (excl. x = 0)
ln(1 + IntegratorToSubmitterLinkIntensity)

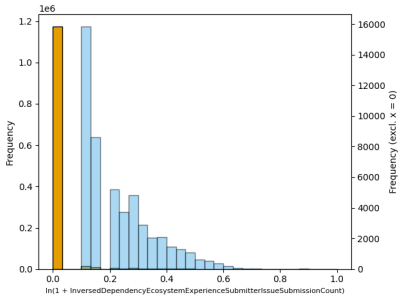

Frequency
Frequency (excl. x = 0)
ln(1 + IntraProjectSubmitterIssueCommentCount)

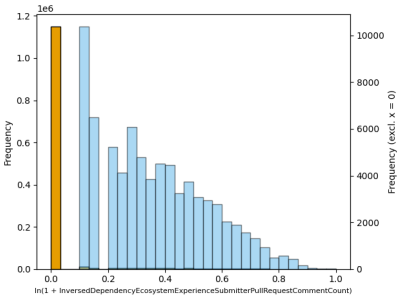

Frequency
Frequency (excl. x = 0)
ln(1 + IntraProjectSubmitterIssueSubmissionCount)

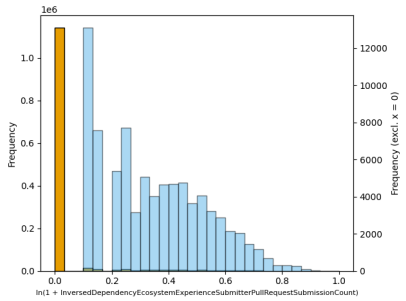

Frequency
Frequency (excl. x = 0)
ln(1 + IntraProjectSubmitterPullRequestCommentCount)

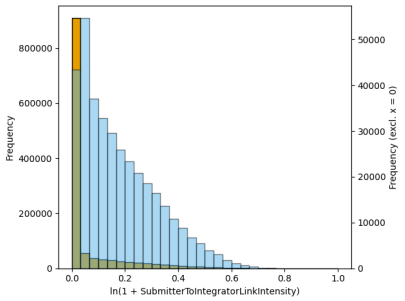

Frequency
Frequency (excl. x = 0)
ln(1 + IntraProjectSubmitterPullRequestSubmissionCount)

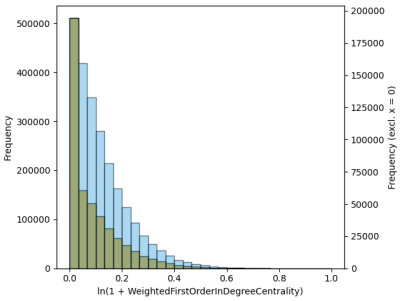

1e6
Frequency
Frequency (excl. x = 0)
ln(1 + InversedDependencyEcosystemExperienceSubmitterIssueCommentCount)

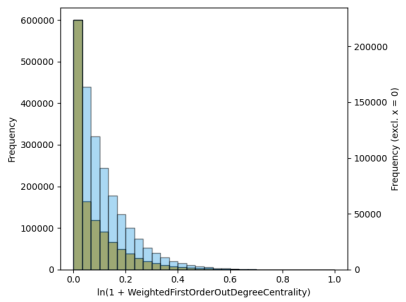

1e6
Frequency
Frequency (excl. x = 0)
ln(1 + InversedDependencyEcosystemExperienceSubmitterIssueSubmissionCount)

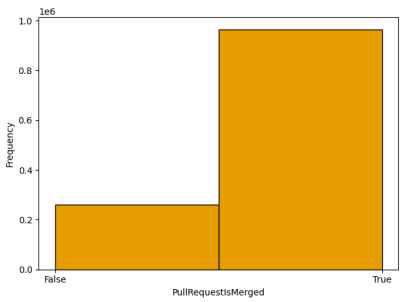

1e6
Frequency
Frequency (excl. x = 0)
ln(1 + InversedDependencyEcosystemExperienceSubmitterPullRequestCommentCount)

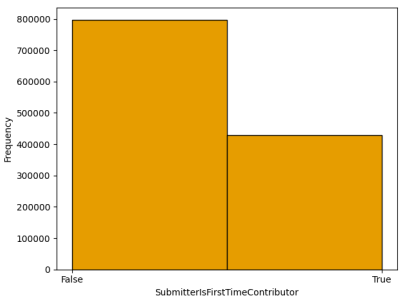

1e6
Frequency
Frequency (excl. x = 0)
ln(1 + InversedDependencyEcosystemExperienceSubmitterPullRequestSubmissionCount)


Frequency
Frequency (excl. x = 0)
ln(1 + SubmitterToIntegratorLinkIntensity)

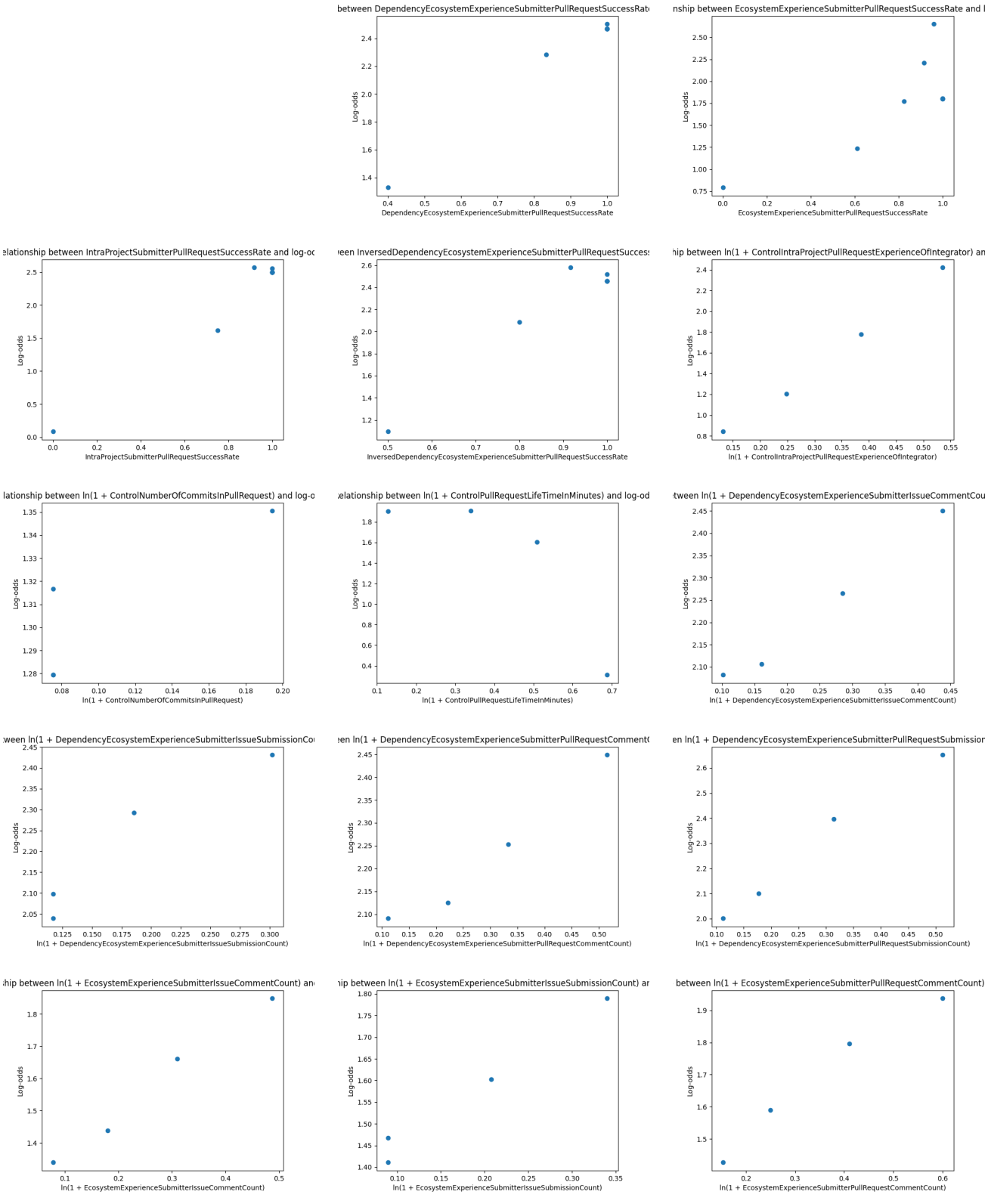

Frequency
Frequency (excl. x = 0)
ln(1 + WeightedFirstOrderInDegreeCentrality)

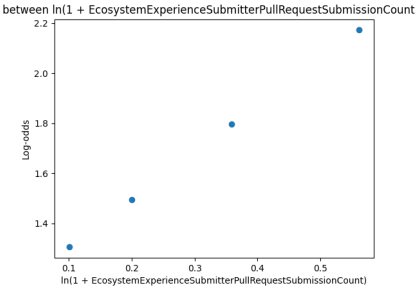

Frequency
Frequency (excl. x = 0)
ln(1 + WeightedFirstOrderOutDegreeCentrality)

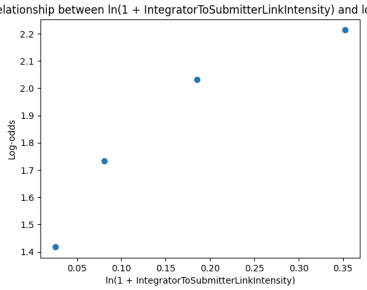

1e6
Frequency
False
True
PullRequestIsMerged

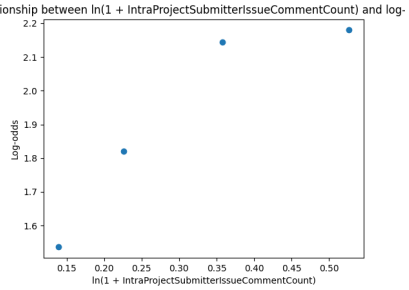

Frequency
False
True
SubmitterIsFirstTimeContributor

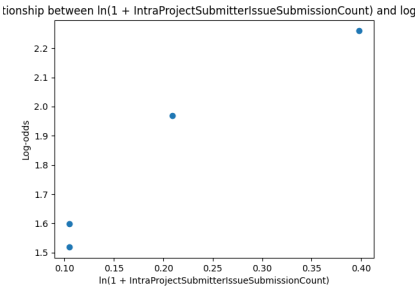
B Variable Log-Linearity

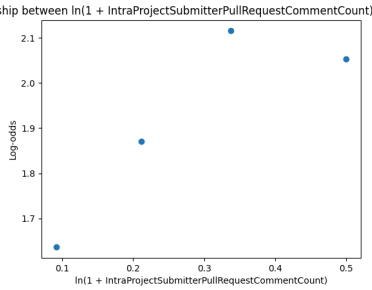


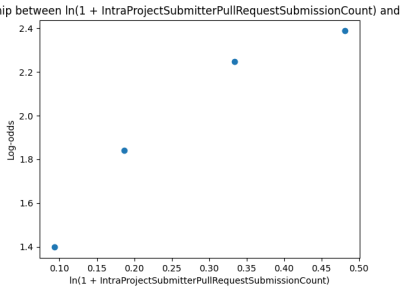

between ln(1 + EcosystemExperienceSubmitterPullRequestSubmissionCount
Log-odds
ln(1 + EcosystemExperienceSubmitterPullRequestSubmissionCount)

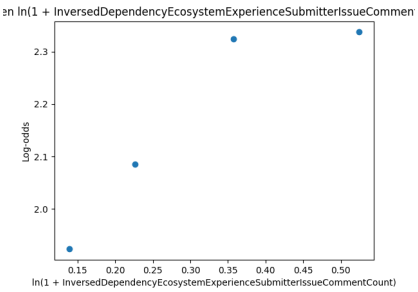

Relationship between ln(1 + IntegratorToSubmitterLinkIntensity) and log-odd
Log-odds
ln(1 + IntegratorToSubmitterLinkIntensity)

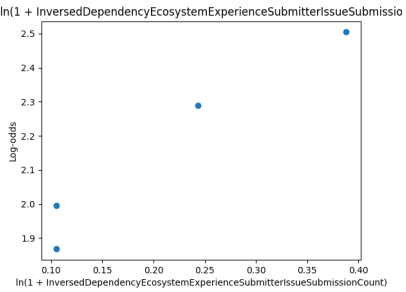

ıtionship between ln(1 + IntraProjectSubmitterIssueCommentCount) and log-
Log-odds
ln(1 + IntraProjectSubmitterIssueCommentCount)

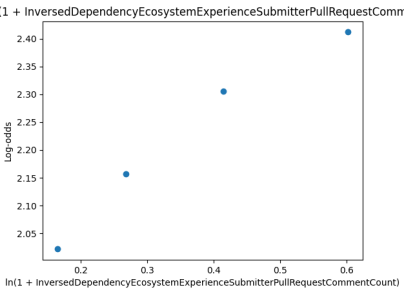

tionship between ln(1 + IntraProjectSubmitterIssueSubmissionCount) and log
Log-odds
ln(1 + IntraProjectSubmitterIssueSubmissionCount)

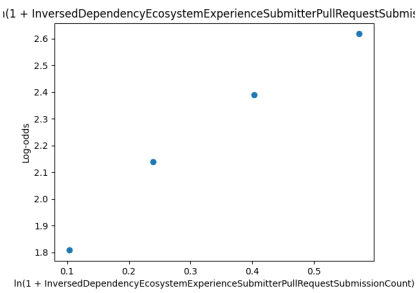

nship between ln(1 + IntraProjectSubmitterPullRequestCommentCount) and l
Log-odds
ln(1 + IntraProjectSubmitterPullRequestCommentCount)

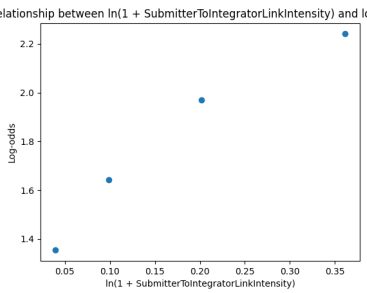

ıship between ln(1 + IntraProjectSubmitterPullRequestSubmissionCount) and
Log-odds
ln(1 + IntraProjectSubmitterPullRequestSubmissionCount)

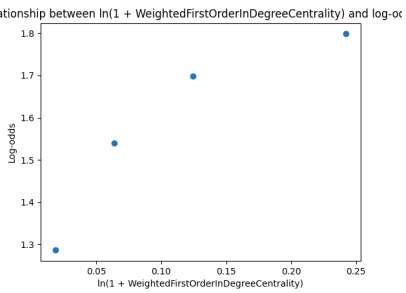

en ln(1 + InversedDependencyEcosystemExperienceSubmitterIssueCommen
Log-odds
ln(1 + InversedDependencyEcosystemExperienceSubmitterIssueCommentCount)

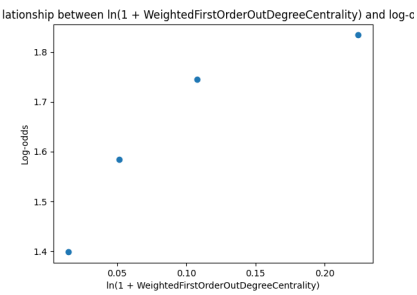

n ln(1 + InversedDependencyEcosystemExperienceSubmitterIssueSubmissio
Log-odds
ln(1 + InversedDependencyEcosystemExperienceSubmitterIssueSubmissionCount)


n(1 + InversedDependencyEcosystemExperienceSubmitterPullRequestComm
Log-odds
ln(1 + InversedDependencyEcosystemExperienceSubmitterPullRequestCommentCount)


ı(1 + InversedDependencyEcosystemExperienceSubmitterPullRequestSubmis
Log-odds
ln(1 + InversedDependencyEcosystemExperienceSubmitterPullRequestSubmissionCount)

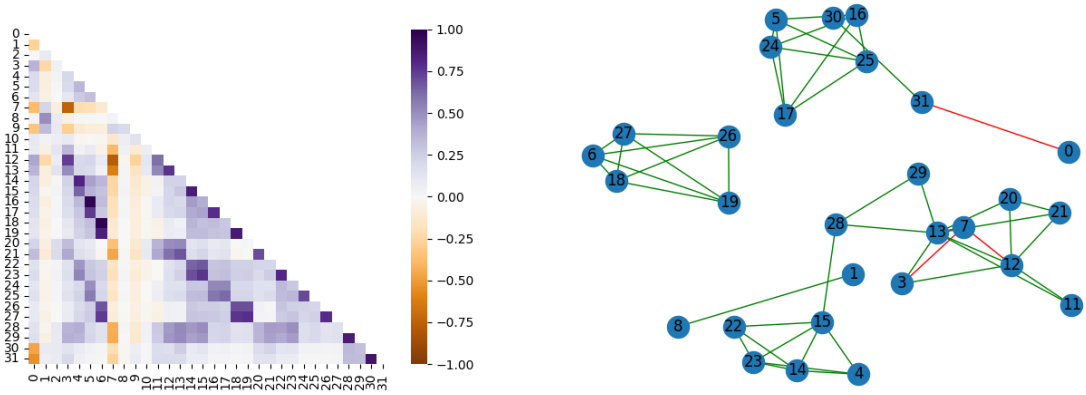

Relationship between ln(1 + SubmitterToIntegratorLinkIntensity) and log-odd
Log-odds
ln(1 + SubmitterToIntegratorLinkIntensity)


elationship between ln(1 + WeightedFirstOrderInDegreeCentrality) and log-oc
Log-odds
ln(1 + WeightedFirstOrderInDegreeCentrality)


lationship between ln(1 + WeightedFirstOrderOutDegreeCentrality) and log-c
Log-odds
ln(1 + WeightedFirstOrderOutDegreeCentrality)

C Variable Correlation and Multicollinearity

In the figures, it is visible that there is a strong correlation between the experience types, as their respective components in the graph are almost always fully connected. It is visible that *"PR has comments"* and *"PR comment from external contributor"* correlate. It is visible that link intensity variables are correlated (30 and 31). The same holds for in-dependency experience (5, 16, 16). Interestingly, these variables negatively correlate with self-integrated pull requests. It is visible that first-order degree variables correlate (28 and 29) and create a bridge between intra-project experience and intra-project experience. Finally, it is visible that being a first-time contributor (7) negatively correlates with intra-project experience (3, 12, 13).



(a) Spearman correlation matrix.

(b) Strongly correlated variables (s.t. $|\rho| \geq 0.5$).

Control variables	
0	PR Self-integrated
1	PR has comments
2	PR has *"#"*
7	First-time contributor
8	PR has comment by external contributor
9	PR lifetime
10	PR commit count
11	Integrator experience

Intra-project experience	
3	Pull request merge rate
12	PR submission count
13	PR comment count
20	Issue submission count
21	Issue comment count

Ecosystem experience	
4	Pull request merge rate
14	PR submission count
15	PR comment count
22	Issue submission count
23	Issue comment count

In-dependency experience	
5	Pull request merge rate
16	PR submission count
17	PR comment count
24	Issue submission count
25	Issue comment count

Out-dependency experience	
6	Pull request merge rate
18	PR submission count
19	PR comment count
26	Issue submission count
27	Issue comment count

Collaborative variables	
28	Weighted first-order in-degree centrality
29	Weighted first-order out-degree centrality
30	Integrator to submitter link intensity
31	Submitter to integrator link intensity