Analyses of software project characteristics on pull request acceptance in distributed software development

Haoyu Wang

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at the University of Glasgow

10th December 10,2021

Abstract

With the continued growth of the web and the advent of distributed version control systems, distributed software development has become a mainstream development approach. Social coding tools such as GitHub [1] have changed how software is developed collaboratively and publicly on the World Wide Web [2]. Instead of pushing changes to a central repository, developers are cloned from other repositories and merged locally [3]. This work builds on the dataset of Zhang et al.'s study [4] to investigate the impact of project features in pull requests on whether pull requests can be successfully merged. There are six main project features, namely Programming languages, Popularity of project, Age of project, Workload of a project, Activeness of project, and Openness of a project. Using the project characteristics provided in the dataset [4], data cleaning, as well as data analysis and data visualization were carried out to find out the relationship between project characteristics and the success rate of pull requests.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

Name: Haoyu Wang

Signature: Haoyu Wang

Acknowledgements

First, I would like to express the deepest appreciation to my supervisor, Handan Gul Calikli, for her all feedback, encouragement, and guidance throughout this project and for this amazing opportunity of studying on Data Analyses. Without her help, I cannot achieve what I have done.

Last but not least, I would like to thank all the participants, who are also my classmates and friends, for taking the time to give me help with the data analysis.

Additionally, I want to thank my parents for their unconditional supports and encouragement.

Contents

Chapter 1 Introduction

GitHub is a hosting platform for open source and private software projects. It provides a platform for software developers around the world to communicate and collaborate with each other. Any software developer can develop and open source their own projects on GitHub. As more and more developers recognize and enjoy distributed development, more and more projects, both open and closed source, are being migrated to code hosting sites like GitHub [3]. GitHub has implemented a distributed development model, called pull requests, for developers to be able to do pull-base development. The special feature of this distributed development model is that it allows any user to clone the public repository uploaded by the development team. This means that the user does not need to be part of the development team to make changes to the repository. Once modified, they can upload their modified code for review via a pull request. If the review is approved, the changes can be merged into the development team's repository, and you can become a contributor to the project.

However, the existence of censorship means that several factors, such as contributor type, project language, development time and many others, can affect the pull request. The study by Zhang et al. [4] builds on the study by Gousios et al. [3] by creating a new upgraded dataset called new_pullreq. This dataset contains a total of 11,230 software projects, 3,347,937 pull requests and 96 features [4]. The 96 features are divided into three broad categories: contributor-related, project-related, and pull request-related. There are features with intersections within the three categories.

The main objective of this MSc project is to conduct an exploratory study by replicating the studies in the literature [5-11] that investigated how project characteristics relate pull request merge success using small datasets. I replicated the mentioned studies using the largest pull requests dataset in the literature [4] to explore to what extent the findings in the literature are generalisable. Empirical investigations I conducted led to the identification of project characteristics that can have a greater impact on pull request acceptance.

The structure of the paper is as follows: Chapter 2 provides background on pull requests, and a review of the studies that investigated how project characteristics related to pull request acceptance. Chapter 3 provides details about the features in the dataset that are about the project characteristics as well as the how the dataset was pre-processed and cleaned. The tools and methods used in the data analyses are also explained in this chapter. Chapter 4 presents the data analyses results. Chapter 5 is a discussion of the data analyses results in the form of the results' implications to software developers who would like contribute to open-source projects on GitHub through pull requests. Chapter 6 concludes with a summary of the whole project and a reflection on the results of future work.

Chapter 2 Background & Related Work

This section provides background information on pull-based software development and pull requests. In recent years, there has been a growing body of research on how project characteristics relate to pull request acceptance. This section also provides a review of these studies in the literature. Review of the related work helped me determine to what extent and how I should replicate their analyses using a dataset that is significantly larger and richer compared to the datasets used in the studies mentioned in this section.

2.1 Pull Request

As described in the introduction, pull request is a distributed software development model that works as follows: first, the core development team uploads their project to the main GitHub repository. When potential contributors want to make changes to the project, they don't do so directly in the main repository, but instead fork a new repository separate from the main one. The contributor can then make changes to the project on this new fork. When the change is complete, it does not appear directly in the main repository, but the contributor generates a pull request and uploads it to GitHub, where it is reviewed by members of the core development team (integrators). If the integrators find the modification satisfactory, the pull request will be merged to the main repository and become a part of the software project; if not, it will be rejected, and the contributor can be asked to make additional changes that will be checked by the integrators if the contributor re-submits. The exact process is shown in Figure 1.

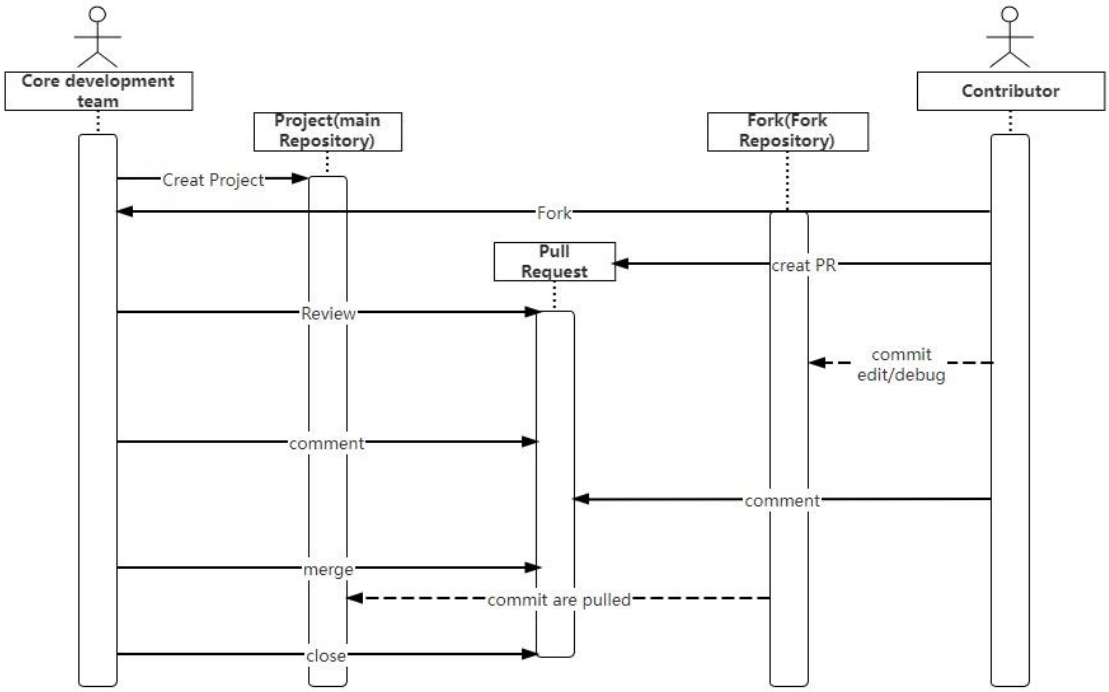


Figure 1 Pull Request Process, adapted from Gousios et al. [8]

2.2 Related work

In a study by Soares et al. [5] on the impact of programming languages on pull request pairs, the authors investigated how the programming languages affect merge lift by using association rules, and by comparing the lift of different programming languages, thus deriving the impact of different programming languages on the success rate of pull request merges. On the other hand, Rahman et al. [6] analysed the distribution of successful and failed merges of pull requests in different languages to derive the of different programming languages in relation to the success rate of merges of pull requests. Padhye al. [7] examined how the number of developers and submissions vary with respect to different programming languages.

Similarly, Rahman et al. [6] investigated the effect of the number of project bifurcations on the merging of pull requests, and examined the effect of bifurcations on merging through a graphical statistical approach. Gousios et al. [8], proposed the use of *watcher count* and *star count* that represent the popularity of projects on GitHub. The main work of Khadke, N et al [2] was to predict the merge outcome of pull requests by using regression models using features such as the number of forks and the number of watchers. Although the authors' aim was different than ours, using regression models for prediction, one can also get information about how the number of forks and the number of viewers correlate to the merge success rate. In contrast, Tsay et al. [9] used the *star count* in a multi-level mixed effects logistic regression model to investigate it correlates the success rate of the merger (pull request acceptance).

Tsay et al. [9] and Yu et al. [10] both used a multi-level mixed effects logistic regression model to predict the likelihood of pull request acceptance. The authors investigated how the age of a project relates to pull request acceptance by estimating the correlation coefficients in the multi-level mixed effects logistic regression models. The multi-level mixed effects logistic regression model by Yu et al. [10] also analysed how project workload relates to pull request acceptance. On the other hand, Baysal et al. [11] first used Kolmogorov-Smirnov tests to examine the data distribution and then used Kruskal-Wallis ANOVA to examine the correlation between project workload and pull request.

Furthermore, Khadke et al. [2] used project activity and openness in logistic regression models to predict the merge success rate of pull requests (pull request acceptance rate). Besides prediction of pull-request acceptance rate, the authors also investigated how project activity and project openness relate to the successful merge of a pull request.

To summarize, these studies demonstrate that pull request acceptance and project characteristics are correlated. However, the studies mentioned above investigate the relation of only a subset of the project characteristics to pull request acceptance on much smaller datasets. In this project, I analyze all project characteristics using the most extensive dataset on pull requests in the literature [4], rich in the variety of software projects included and programming languages used to develop these software projects.

Chapter 3 Tools and Methodology

3.1 Dataset

The dataset used in this study is an upgraded version of the dataset created by Zhang et al. [4], Table 1 shows the basic information in the dataset.

Dataset	Project	Feature	Pull Request
Total	11,230	96	3,347,937

Table 1 Dataset properties

From the table we can see that the dataset contains a total of 3,347,937 pull requests, which come from a total of 11,230 software development projects and each pull request has 96 features. Zhang et al. [4] categorised 96 features as project characteristics, contributor characteristics and pull request characteristics, respectively. In this study, I focus on the project characteristics, which contain a total of seven features in the dataset, grouped into six project characteristics. Table 2 shows the features included in the project characteristics and the corresponding features in the dataset.

***Project* Characteristics**		
Project characteristics	Feature	Description
Programming language	*Language*	Languages used in the project [4]
Popularity of Project	*fork_num*	Number of branches that existed in the project at the time of the pull request [4]
Age of Project	*project_age*	Time interval between pull request creation and project creation [4]
Workload of Project	*open_pr_num*	The number of pull requests that were being reviewed at the time the pull request was uploaded. [4]
Activeness of Project	*pushed_delta*	The time interval between the two most recent pull request requests opened. [4]

Openness of Project	*open_issue_num*	Number of issues already open when the pull request is submitted [4]
Another	*merged_or_not*	Determining whether a pull request merge is successful [4]
	core_member	Determine if the person submitting the pull request is a core member [4]
	pr_succ_rate	Pull request acceptance rate for this project at the time of pull request submission [4]

Table 2 The features of the projects contained in the dataset and the interpretation of the features to be used.

Descriptive Statistics

Firstly, a descriptive statistic is made of the required data, as shown in Table 3.

Dataset Statistics						
	Mean	Std	Min	Mid	Max	Count
project_age(month)	33.87557	24.40073	-22	30	135	3347937
pushed_delta(seconds)	295904.3	1339993	0	27414	157351000	3327395
pr_succ_rate	0.96851	0.049237	0	0.979	1	3337734
open_pr_num	86.4924	314.88449	0	13	10936	3347937
open_issue_num	251.1323	754.88578	0	32	7263	3347937
fork_num	768.346	2231.8421	0	78	34664	3347937
merged_or_not	0.809	0.393	0	1	1	3347937

Table 3 Summary of our dataset before removing outliers

We can see from Table 3, on average projects are close to 3 years old (2.8 year more precisely) and oldest software project is slightly older than 10 years old (~ 11 years old). Here we find that the smallest item creation time is -22. After reviewing the

relevant paper by Zhang et al.[4] on the description of the dataset, there is no explanation for such negative values in the dataset, and I believe that this may be due to an error in the process of crawling the pull request, which led to this situation. And I will explain the handling of such outliers in the following data processing. We can also get from the table,mean pull request success rate (pr_succ_rate) being 0.968 is quite high. Seems like pull request acceptance rates are quite high in the projects and a small portion o the pull request are not accepted. Looking at the time interval between the opening of two pull requests (pushed_delta), we can see that the average time interval is 83 minutes, which means that it basically takes almost 83 minutes for a pull request to be reviewed.

Data Processing

We pre-processed the dataset as follows:

1) As a result of examining values of each feature in the dataset, I detected that there are outliers in the values of the feature *project_age* (i.e., age of asoftware project) and the smallest value is -22 months, which is not possible (see Table 3). There are also 341 values in *project_age* that are less than 0. We removed from the dataset 341 pull-requests where age of the corresponding software project is negative. Since, the dataset consists of more than 3 million (precisely 3,347,937) pull-requests, deleting 341 data items does not significantly affect the analyses results.
2) In order to handle missing data in the values for the features pushed_delta and pr_succ_rate, I chose to fill in the missing data by the mean values.

3.2 Data analysis

As a first step I conducted Kolmogorov-Smirnov (KS) test [19] to test whether each of the feature values are normally distributed or not. KS test results for all features in addition to programming languages indicate that data for none of the features is normally distributed (i.e., $p\text{-}value < 0.05$). Chi-square test results for language, *fork_num, project*_age and *merged_or_not*. A $p\text{-}value < 0.05$ was found, indicating that these three items were correlated with pull request acceptance (i.e., whether the pull request was merged or not). In order to analyse how *project_work, project_openness* and *project_activity* relate pull request acceptance, I used a regression model where the correlation coefficients are estimated Below separate analysis of each of the project characteristics individually are explained in more detail,

3.2.1 Programming languages

The dataset by Zhang et al. [4] contains six programming languages,wihich are JavaScript, Java, Python, Ruby, Go and Scala, where different projects use different subsets of these languages and receiving different rates of pull requests. The metrics in the dataset are *languages* [4, 5, 6, 7].

Descriptive Statistics

The statistics for the 11,230 projects in the dataset are shown in Figure 1, which shows that JavaScript, Java, and Python are used more frequently in all projects. making up almost 80 per cent of the projects.

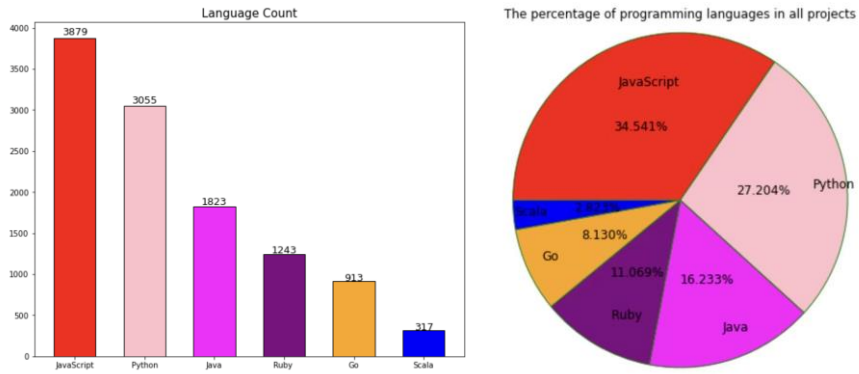


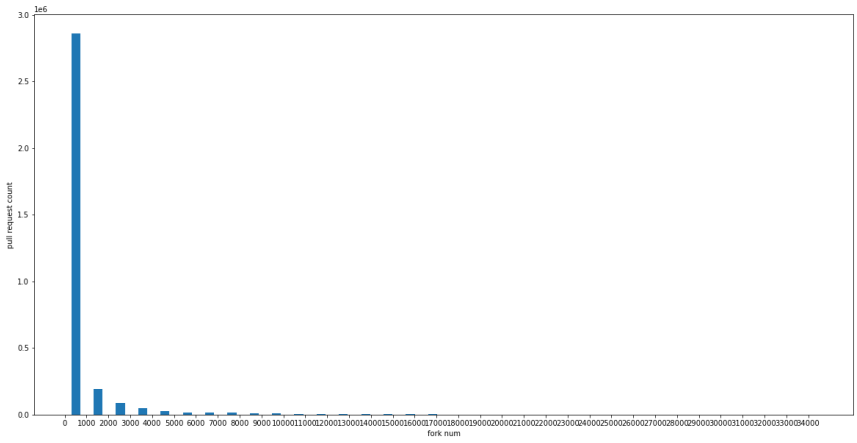
Figure 2 Programming language count and Percentage pie chart

3.2.2 Popularity of project

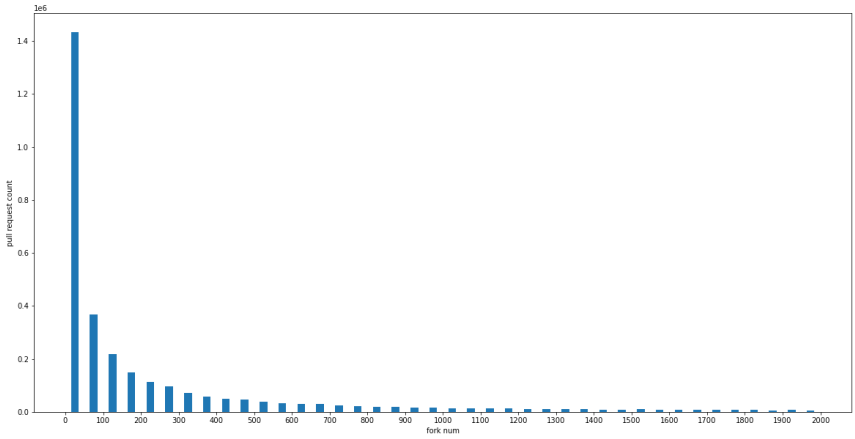
The popularity of a project is mainly determined by the number of forks that already exist in the project at the time of the pull request, the higher the number of forks, the more popular the project is. This is represented by *fork_count* [4] in the dataset metrics.

Descriptive Statistics

The number of forks of the project at the time of all pull requests is shown in Figure 3(a), and most of the forks of the project at the time of the pull request are within 1000 forks, and a few are within 1000 to 2000 forks, and even fewer are greater than 2000. To get a more accurate value, I show the distribution of the number of pull request forks within 2000, as shown in Figure 3(b), and find that most of the pull request uploads are within 100 forks, and then gradually decreasing.



(a)



(b)

Figure 3 (a) and (b), statistics on the distribution of the number of forks in a project when a pull request is uploaded. (a) is the distribution of the number of all forks. (b) is the number of forks less than 2000

3.2.3 Age of project

The age of a project is the interval between when the project was created and when the pull request was created (month) and is represented in the dataset metrics using *project_age* [4].

Descriptive Statistics

A histogram of the distribution of this data across all data sets, as shown in Figure 4, shows that the data is relatively evenly distributed, decreasing as the age of the projects increases.

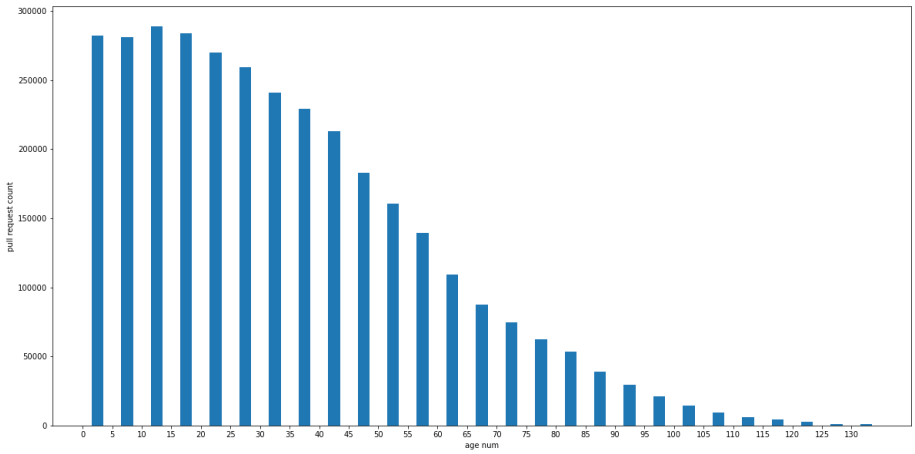


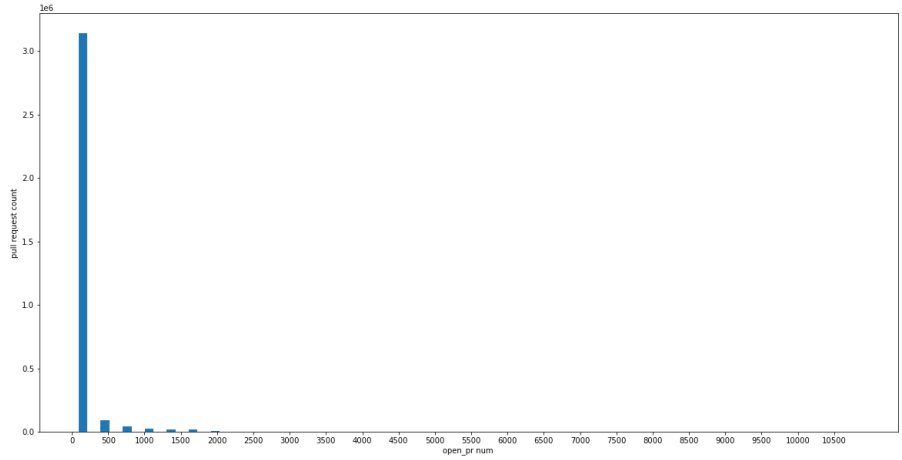
Figure 4 Age distribution of projects when pull requests are submitted

3.2.4 Workload of project

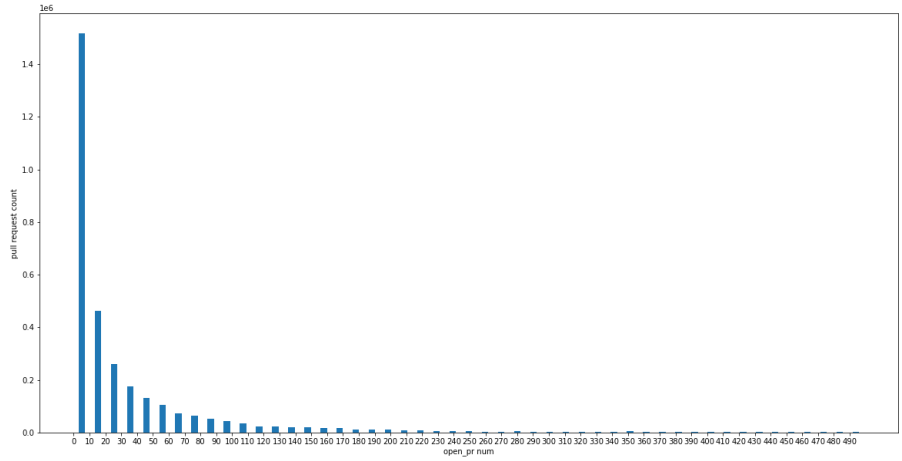
The workload of a project is the number of pull requests in the queue waiting to be reviewed when the pull request is submitted, expressed as *open_pr_num* [4] in the dataset metrics. A count of this feature in the dataset is performed to see the distribution of this data.

Descriptive Statistics

As shown in Figure 5(a), most of the data is concentrated between the number of pull requests 0-500, so look again at the distribution within 0-500, as shown in Figure 5(b). It can be found that most of the data is still concentrated between 0-10.



(a)



(b)

Figure 5 (a) and (b) Distribution of the number of pull requests opened by the project when a pull request is uploaded (a) is the distribution of all (b) is the distribution of the number of pull requests opened by the project that are less than 500.

3.2.5 Activeness of project

The activeness of a project is measured by the interval between two pull requests when the last two pull requests for that project were uploaded (seconds). The shorter the time interval the more active the project is. It is represented by the fature *pushed_delta* [4] in the dataset.

3.2.6 Openness of a project

The openness of the project is the number of issues that exist at the time of the pull request's submission. The higher the number of issues, the higher the openness. The feature *open_issue_num* [4] is used in the dataset as a measurement of a project's opennes.

3.2.7 Another feature

The remaining three features are three related features that need to be used in the replication paper, namely *merged_or_not* [4], *core_member* [4] and *pr_succ_rate* [4]. The feature *merged_or_not* indicates whether the pull request is merged, and *core_member* is used to determine whether the member submitting the pull request is the core member of the software project. Finally, *pr_succ_rate* is the pull request acceptance rate of the project at the time of the pull request's submission.

3.3 Data analyses techniques

This section describes statistical models and tests used in the data analyses.

3.3.1 Logistic Regression model

Since the problem is dichotomous (i.e., a pull request is accepted or not), I used a logistic regression model to predict the outcome of a pull request given the features concerning project characteristics. The aim is to see how each of the project related features correlate with the pull request acceptance. Scikit-learn [12] (a machine learning library for Python) was chosen to implement the logistic regression models to predict a pull request's outcome (i.e., whether pull request is accepted or not). I chose to use two logistic regression models, which are with a L1 penalty term and a L2 penalty term, respectively in the analyses.

3.3.2 Multi-level Mixed Effects Logistic Regression Model

In this study I also used multiple linear regression to model the likelihood of pull request acceptance and thus determine the correlation between the different feature. The model I implemented uses lme4 [13], which is originally a package in the R language, but we was able to use the lme4 model on python for prediction by using Pymer4 [14].

3.4 Association Rules

Association Rules is a technique in data mining that reflects the interdependence and association between one data item and other data items and is used to extract valuable relationships between data items from large amounts of data. In this study, multidimensional association rules were used to analyse the data [15]. An association rule is an implication of $X \rightarrow Y$, where X is the prior of the association rule and Y is the successor of the association rule. In layman's terms we can explain an association rule $X \rightarrow Y$ as follows: X is the cause of an event, Y is the effect of the event, and the presence of X implies the presence of Y. The relevance of association rules is assessed by two main indicators of interest: support and confidence [16]. Suppose there exists a dataset D. D is the set of transactions and contains N transactions. The support is the percentage of transactions in D that contain both X and Y, i.e., the probability; the confidence is the percentage of transactions in D that contain Y if they already contain X, i.e., the conditional probability. An association rule is considered interesting if a minimum support threshold and a minimum confidence threshold are satisfied [17].

Another metric, lift, is introduced to express the correlation between X and Y. The formula for lift is $\text{Lift}(X \rightarrow Y) = \text{Conf}(X \rightarrow Y)/\text{Sup}(X)$ [5]. When lift is 1, it means that X and Y are independent of each other, and when lift is greater than 1, it means that X and Y are positively correlated with each other. When lift is less than 1, it means that X and Y are negatively correlated.

In this study the Apriori algorithm was chosen for association rule acquisition. We used MLXTEND [18] (a python library for everyday data science tasks) for implementation.

Chapter 4 Results

4.1 Programming languages

To analyse how programming languages relate to pull request acceptance, I examined the distribution of successful and merged pull requests for each programming language replicating the analyses Rahman et al. [6] had conducted on a smaller dataset The statistics are shown in Table 4 and Figure 6.

LANGUAGE	MERGE FAILED	MERGED	ALL	MERGE FAILED PERCENT	MERGED PERCENT
GO	41706	239477	281183	0.148	0.852
JAVA	139836	513864	653700	0.214	0.786
JAVASCRIPT	178134	815182	993316	0.179	0.821
PYTHON	172645	751833	924478	0.187	0.813
RUBY	72569	298472	371041	0.196	0.804
SCALA	32951	91268	124219	0.265	0.735
ALL	637841	2710096	3347937	0.191	0.809

Table 4 Programming language merge success or failure statistics table

For a more visual display, we have taken a picture to show these data, as shown in Figure 6.

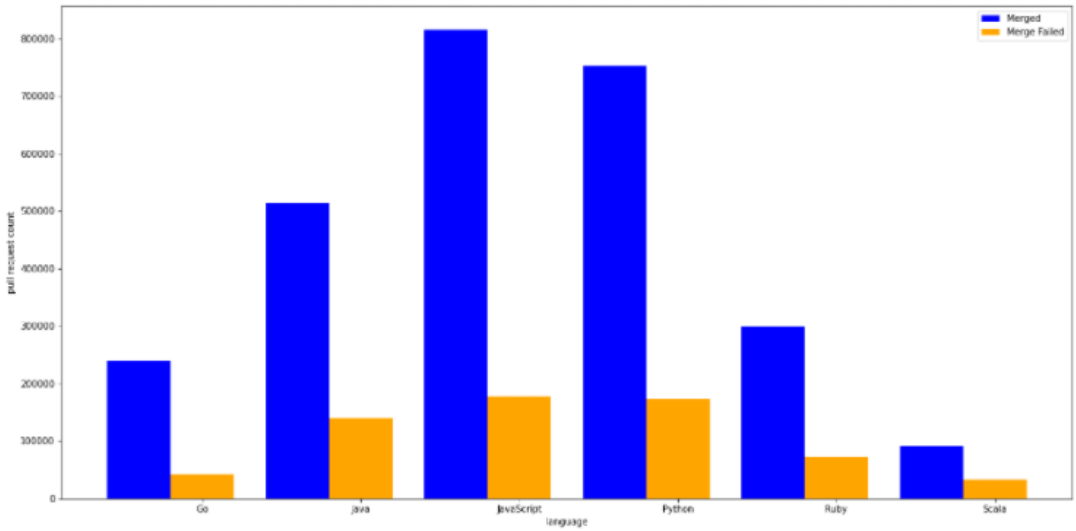


Figure 6 Project programming language pull request merge statistics chart

As shown in Figure 6, pull requests are more likely to be accepted for projects using Go, JavaScript, and Ruby, and Python, while the opposite is true for Java, and Scala. Another way to analyse the data is to use association rules to calculate the lift, as Soares D M et al. [5] did, to see the correlation between each language and the pull request merge. As shown in Figure 7, it can be seen that although the difference in lift between each language is small, it can still be seen that Go, Python and JavaScript have a lift > 1, indicating that they are positively correlated with pull request acceptance, while Java and Scala have a lift < 1, indicating that they are negatively correlated. This is the same result as the one we obtained earlier in the statistics. Here, the reason why the conclusions obtained by Soares D M et al. [5] and Rahman M et al. [6] are biased is because the datasets used in their study and this study are different. There is a significant difference between their dataset and the one used in this study in terms of the number of items and the number of pull requests, and their dataset contains a total of 13 languages while the present dataset contains only 6 languages. This may be the reason for the bias in the results.

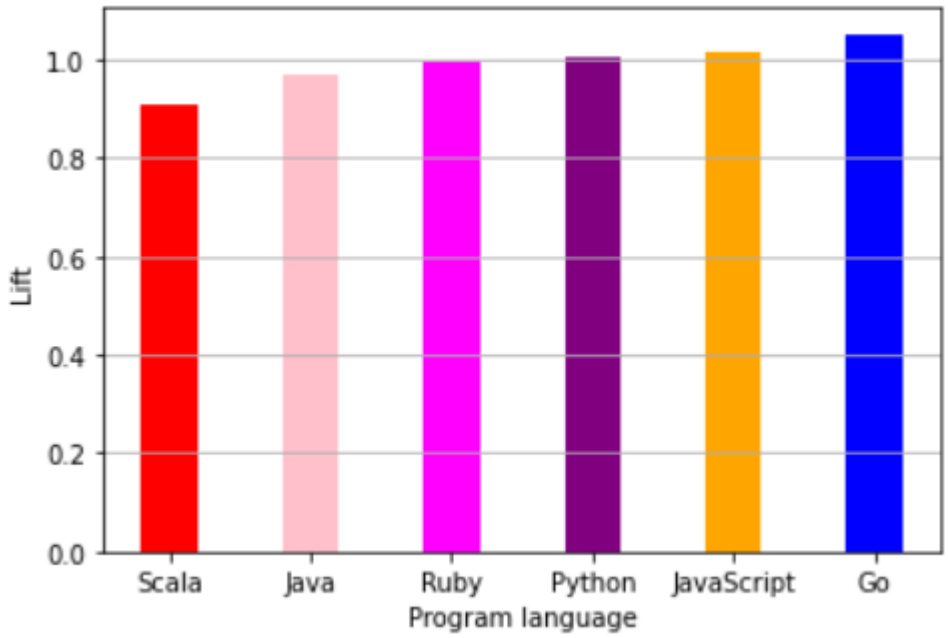


Figure 7 Lift of the rule: language→ merge

The article by Padhye R et al. [7] is not replicated in this study because in their paper, the statistics are between languages and developers, and in his dataset, there are three types of developers: CORE committer, EXTERNAL commit, and MUTANT, but there is only one feature in our dataset to determine if a core person submitted the pull request, so it is not possible to replicate their article.

4.2 Popularity of project

The analysis of the popularity of the items was carried out using the same two methods of analysis.

The first also uses the same statistical analysis as the study by Rahman M et al. [6]. From the analysis in Figure 3(a) in Chapter 3 on item popularity, it is known that most of the data is concentrated in the 0-1000 range, so further refinement of the data in the 0-1000 range is shown in Figure 3(b), where the data is divided into nine stages to investigate the relationship between item popularity better and pull request acceptance, as shown in Table 5.

FORK NUM	MERGE FAILED	MERGED	ALL	MERGE FAILED PERCENT	MERGED PERCENT
0-50	68325	1242840	1442529	0.138	0.862
50-100	41670	300934	364184	0.174	0.826
100-200	49796	295282	363607	0.188	0.812
200-300	61055	168010	20968	0.199	0.801
300-500	56344	174454	22425	0.222	0.778

500-1000	26344	187417	248472	0.246	0.754
1000-2000	71368	135678	192022	0.293	0.707
2000-3000	637841	63529	89873	0.293	0.707
>3000	68325	141952	213320	0.335	0.665
ALL	41670	2710096	3347937	0.191	0.809

Table 5 Popularity of project merge success or failure statistics table

It is already clear from the table that as the number of project forks increases, i.e. as the project becomes more popular, the more likely it is that the merge will fail. To visualise this result, even more, we have presented the data in a bar chart. As shown in Figure 8.

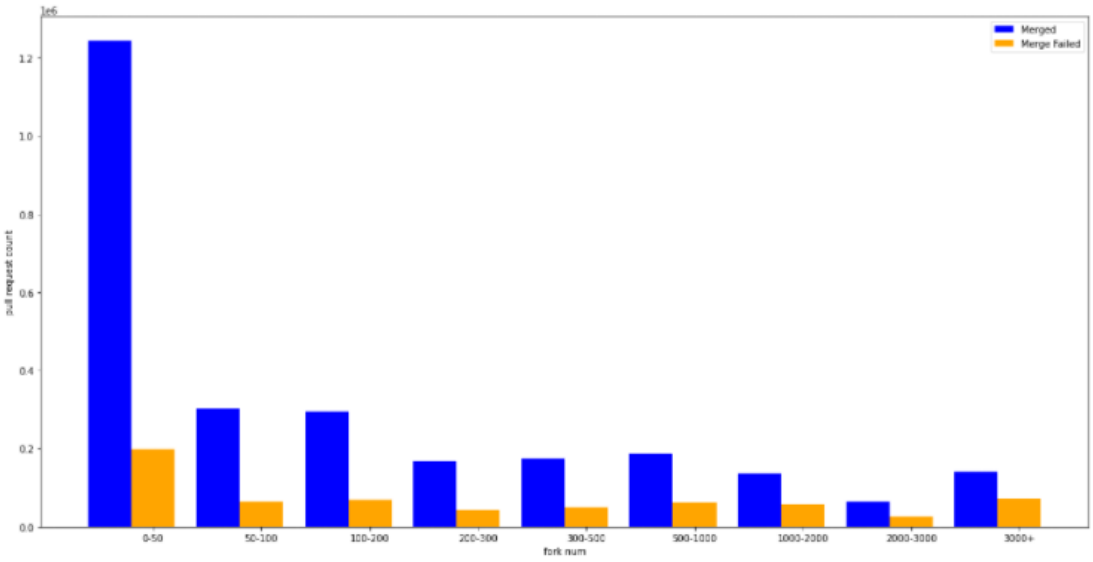


Figure 8 Popularity of project pull request merge statistics chart

In the study by Rahman M et al. [6] , the analysis also shows that the higher the number of forks, the higher the failure rate of pull request merges, proving our analysis correct.

Another approach is to use a logistic regression model to find the correlation between the number of forks and the merge, which will be analysed in the following subsection. The part using the logistic regression model is done in conjunction with the openness and activity characteristics of the project.

The study by Jason Tsay et al. [9] is not replicated in this section because in having them multiply, the study used STARS to represent the popularity of the items. Still, our dataset does not contain this one feature. Therefore, it is not possible to replicate their pap

4.3 Activeness of project and Openness of project (Including the popularity of the project)

In this section, a logistic regression model[2] is used to train the openness, activity and popularity of items in the dataset to obtain a trained model, and the correlation between these three features and the pull request merge is obtained by looking at the coefficients of the features obtained after training. Table 6 depicts the model accuracy and f1 scores obtained using the l1 penalty term and l2.

Regularization Metric	Mean Accuracy	Std Dev	Mean F1	Std Dev	AUC
L1	0.808	0.000175	0.893	0.000102	0.608
L2	0.808	0.000175	0.893	0.000102	0.608

Table 6 Results of Logistic Regression on initial feature set

There is little difference between the two models in terms of accuracy, f1 scores and AUC. The weights of the different features in the two models were then examined and the weights are shown in Table 7.

Feature	L1	L2
pushed_delta	-0.00424368	-0.00423981
pr_succ_rate	0.23072823	0.23073092
open_issue_num	0.08265489	0.08271627
fork_num	-0.2360897	-0.23612521

Table 7 Weights for different features in L2 regularized Logistic Regression and in L1 regularized Logistic Regression.

It can be noticed that this is negatively correlated for pushed_delta and fork_num, indicating that as the number of forks increases, the merge failure rate becomes higher, which is the same conclusion reached in the previous subsection. In addition, the failure rate increases with the time interval between the last pull request of the project. Unlike these two, the success rate of merging increases as the number of unresolved issues in the project increases and the acceptance rate of the project pull request increases. The results obtained are consistent with the findings of Nikhil Khadke et al. [2]. Although we both used different datasets, as the results obtained are consistent, they suggest that these four characteristics have a consistent impact on the success rate of pull request mergers.

4.4 Age of project and Workload of project

The literature by Rahman M et al. [6] does not contain a statistical analysis of the effect of project age and project workload on the success rate of pull request mergers. Although in this study, it was decided to use their methodology first. First, as with the other two characteristics, the data were statistically analysed. Here, both project age and project workload were divided into 14 ranges, and the statistical information is shown in Tables 8 and 9 respectively.

AGE	MERGE FAILED	MERGED	ALL	MERGE FAILED PERCENT	MERGED PERCENT
0-10	100724	518466	619190	0.163	0.837
10-20	110471	460875	571346	0.193	0.807
20-30	102008	421064	523072	0.195	0.805
30-40	90140	308645	464888	0.194	0.806
40-50	75826	308645	384471	0.197	0.803
50-60	59948	231256	291204	0.206	0.794
60-70	37545	151072	188617	0.199	0.801
70-80	26890	106213	133103	0.202	0.798
80-90	17281	70840	88121	0.196	0.804
90-100	9407	37789	47196	0.199	0.801
100-110	4438	17432	21870	0.203	0.797
110-120	2366	7786	10152	0.233	0.767
120-130	658	3132	3790	0.174	0.826
>130	98	478	576	0.170	0.830
ALL	637800	2709796	3347596	0.191	0.809

Table 8 Age of project merge success or failure statistics table

OPEN PR NUM	MERGE FAILED	MERGED	ALL	MERGE FAILED PERCENT	MERGED PERCENT
0-10	217358	1299715	1517073	0.143	0.857
10-20	84831	377415	462246	0.184	0.816
20-30	52296	208825	261121	0.2002	0.800
30-40	37065	137905	174970	0.212	0.788

40-50	28756	102738	131494	0.219	0.781
50-100	75679	260827	336506	0.225	0.775
100-200	49449	147907	197356	0.251	0.749
200-300	13709	43420	57129	0.240	0.760
300-400	12274	24418	36692	0.335	0.665
400-500	12862	17330	30192	0.426	0.574
500-1000	31021	46089	77110	0.402	0.598
1000-1500	13057	21108	34165	0.382	0.618
1500-2000	7144	14576	21720	0.329	0.671
>2000	2340	7823	10163	0.230	0.770
ALL	637841	2710096	3347937	0.191	0.809

Table 9 Workload **of project merge success or failure statistics table**

From Table 8, it can roughly see that the probability of a successful merge decreases as the age of the project increases, which may mean that as the project matures, it requires fewer and more minor changes or adjustments from contributors and therefore causes project developers to accept less and less pull requests. Looking at Table 9 again, we can see that as the project's pull request open queue increases, the merge success rate decreases. This could be due to the fact that as the number of pull request requests increases, the core developers assign priorities to the process of evaluating pull request requests, the higher the ranking of the pull requests in the queue the more likely they are to are likely to be accepted because they are likely to solve a problem. Pull requests at the bottom of the queue are likely to be rejected because they solve the same problem as the previous pull request, and therefore are not accepted because the issue has already been solved. To make the data more intuitive, we present the data from these two tables using histograms, as shown in Figures 9 and 10.

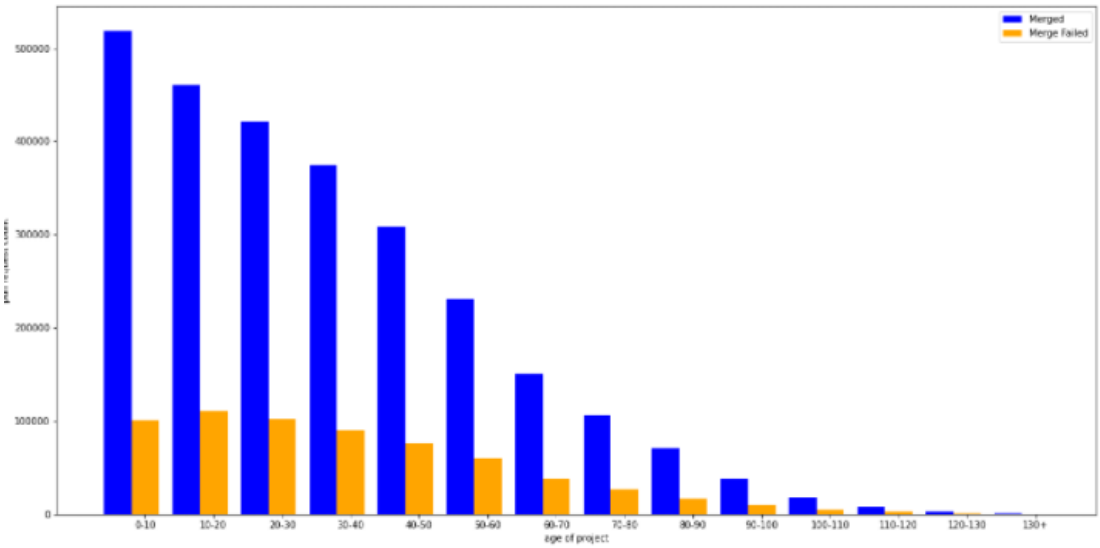


Figure 9 Age of project pull request merge statistics chart

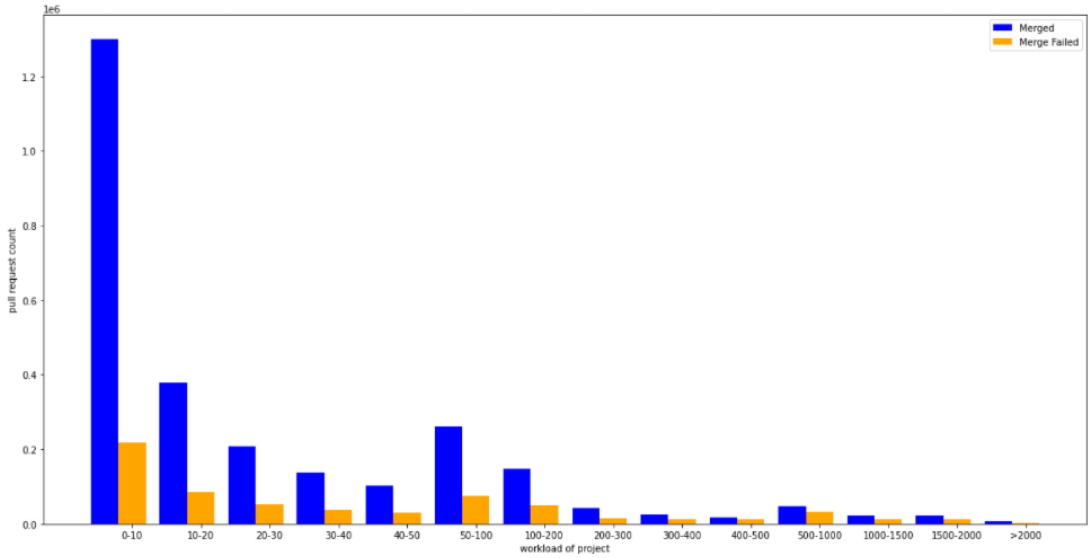


Figure 10 workload of project pull request merge statistics chart

To further demonstrate our results, I used a multilevel mixed-effects logistic regression model to calculate coefficients and significance levels based on the approach described in the article <<Determinants of pull-based development in the context of continuous integration>>[10]. This is because the results for each pull request are dichotomous (i.e., *merged_or_not*).

	Estimate	2.5_ci	97.5_ci	SE	DF	T-stat	P-val	Sig
(Intercept)	0.786	0.782	0.790	0.002	11827.331	432.120	0.0	***
project_age	0.002	0.002	0.002	0.000	3045905.163	179.354	0.0	***
open_pr_num	-0.000	-0.000	-0.000	0.000	3300297.142	-20.017	0.0	***

Figure 11

After simulation using a multilevel mixed effects logistic regression model, as shown in Figure 11, it can be found that the p-value of the results obtained for both, regardless of the workload of the project or the age of the project, is less than 0.05, which proves that these two characteristics are significantly correlated with pull request acceptance rates. Looking again at the estimate obtained for both of them, it can be found that on project age, the estimate obtained is positive, which indicates that as the age of the project increases, the merge success rate also increases, which is not in line with the conclusion we reached above but is consistent with the results obtained in the article by Yue YU et al. I think this may be due to the difference in the dataset, in their article suggest that as the project ages, the project becomes more mature. The project becomes more open. They are using the age of the project as an indicator of project openness. In contrast, we are using the problems with the project as an indicator of project openness. The estimated project workload is negative, suggesting that the success rate of pull request merging decreases as the workload increases in the available dataset, which is consistent with our results and with the results obtained by Yue Y et al. However, the results obtained are inconsistent with those of Baysal O et al.[11] who suggest that as workload rises, pull requests are more likely to be merged.

Chapter 5 Discussion

In this study, we analysed 3347937 pull requests from 11230 projects. We analysed the project factors that influence the merging of pull requests in pull requests. This section discusses and summarises the findings of the study.
An analysis of six project characteristics of pull requests has led to the following insights that may improve acceptance rates when developers submit a pull request：

- More commonly used and widely available languages such as JAVASCRIPT, PYTHON, and GO are more likely to accept pull requests for project development languages.
- For a project, when the project has more forks, it means that the project is more popular, and often popular projects accept more pull request requests. More pull request requests lead to a rise in competitiveness among contributors, which results in pull requests not being easily accepted when they are submitted.
- When contributors select a project for development, they should choose a project with a high level of activity and a relatively high number of problems, as a high level of activity proves that the project's core development team will then actively review each pull request and the higher the number of problems the project has, the more the project will need the help of other developers and the higher the acceptance rate of pull requests.
- When selecting projects for development, contributors should avoid projects that have been created for a long time, as the older the project, the more mature it is, and mature projects often have problems that have already been solved, making it difficult for contributors to get their pull requests accepted.

- Contributors should submit pull requests as early as possible, as they are submitted by someone other than the project developer, often without direct contact between the contributor and developer, resulting in inconsistencies

in the understanding of the code between the two, and multiple submissions will result in lower priority in the pull request queue, thus reducing the number of accepted chances of acceptance.

Chapter 6 Conclusion and future work

In this study, we used various methods to correlate the factors influencing the acceptance of pull requests. Still, as only seven of these features were selected for analysis in a dataset containing 96 features that were relevant to the project, it is possible that the results obtained may not be complete. In the future, I hope to spend more time going through all the features to do an analysis, combining the different features to find out the important factors that affect the acceptance of pull request and provide a suggestion to the contributors that can improve the chance of pull request acceptance.

References

[1] GitHub: Social Coding. [Online] Available: http://github.com/

[2] Khadke, N., Teh, M. H., & Shen, M. (2012). Predicting acceptance of GitHub pull requests. Stanford–CS 229, Tech. Rep.

[3] Gousios, G., Pinzger, M., & Deursen, A. V. (2014, May). An exploratory study of the pull-based software development model. In Proceedings of the 36th International Conference on Software Engineering (pp. 345-355).

[4] Zhang, X., Rastogi, A., & Yu, Y. (2020, June). On the Shoulders of Giants: A New Dataset for Pull-based Development Research. In Proceedings of the 17th International Conference on Mining Software Repositories (pp. 543-547).

[5] Soares D M, de Lima Júnior M L, Murta L, et al. Acceptance factors of pull requests in open-source projects[C]//Proceedings of the 30th Annual ACM Symposium on Applied Computing. 2015: 1541-154

[6] Rahman M, Roy C K. An insight into the pull requests of GitHub[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. 2014: 364-367.

[7] Padhye R, Mani S, Sinha V S. A study of external community contribution to open-source projects on GitHub[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. 2014: 332-335.

[8] Gousios G, Zaidman A. A dataset for pull-based development research[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. 2014: 368-371.

[9] Tsay J, Dabbish L, Herbsleb J. Influence of social and technical factors for evaluating contribution in GitHub[C]//Proceedings of the 36th international conference on Software engineering. 2014: 356-366.

[10] Yu Y, Yin G, Wang T, et al. Determinants of pull-based development in the context of continuous integration[J]. Science China Information Sciences, 2016, 59(8): 1-14.
[11] Baysal O, Kononenko O, Holmes R, et al. The influence of non-technical factors on code review[C]//2013 20th working conference on reverse engineering (WCRE). IEEE, 2013: 122-131.
[12] Scikit-Learn [Online] Available: http://scikitlearn.org/
[13] Bates D M. lme44: mixed-effects modeling with R. 2010. http://lme4.r-forge.r-project.org/lMMwR/lrgprt.pdf
[14] Jolly E. Pymer4: connecting R and Python for linear mixed modeling[J]. Journal of Open Source Software, 2018, 3(31): 862.
[15] Han J, Pei J, Kamber M. Data mining: concepts and techniques[M]. Elsevier, 2011.
[16] I. H. Witten, E. Frank, and M. A. Hall. Data Mining: Practical Machine Learning Tools and Techniques. Elsevier, Feb. 2011.
[17] Jiang Shengyi, Li Xia, Zheng Qi. Principles and practice of data mining [M]. Dian zi gong ye chu ban she, 2011.
[18] MLXTEND [Online] Available: http://rasbt.github.io/mlxtend/
[19] Massey Jr F J. The Kolmogorov-Smirnov test for goodness of fit[J]. Journal of the American statistical Association, 1951, 46(253): 68-78
[20] Chi-squared test: https://en.wikipedia.org/wiki/Chi-squared_test

Appendix A

First appendix

A.1 Useful Links

“new_pullreq” Dataset

https://zenodo.org/record/3922907#.YV6aQRBBwUE

Run Linear Mixed Effects Models in Python Jupyter Notebooks

https://towardsdatascience.com/how-to-run-linear-mixed-effects-models-in-python-jupyter-notebooks-4f8079c4b589

Chi-Square Test for Independence in Python

https://towardsdatascience.com/chi-square-test-for-independence-in-python-with-examples-from-the-ibm-hr-analytics-dataset-97b9ec9bb80a

A.2 My Code

GitHub include code and relate work

https://github.com/vitaminvaf/MSc-Project