

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, especially as social coding tools such as GitHub[1]have changed the way software is developed collaboratively and publicly on the World Wide Web [2]. Instead of pushing changes to a central repository, development developers are pulling them from other repositories and merging them locally [3]. This work builds on the dataset of Xunhui Zhang, Ayushi Rastogi, Yue Yu 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 2

Chapter 2 Background & Related Work 3

2.1 Pull Request 3

2.2 Relate work 3

Chapter 3 Tools and Methodology 5

3.1 Dataset 5

3.2 Data analysis 7

3.2.1 Programming languages 7

3.2.2 Popularity of project 8

3.2.3 Age of project 9

3.2.4 Workload of project 9

3.2.5 Activeness of project 10

3.2.6 Openness of a project 11

3.2.7 Another feature 11

3.3 Regression model 11

3.3.1 Logistic Regression model 11

3.3.2 Multi-level Mixed Effects Logistic Regression Model 11

3.4 Association Rules 11

Chapter 4 Results 13

4.1 Programming languages 13

4.2 Popularity of project 15

4.3 Activeness of project and Openness of project(Including the popularity of the project) 16

4.4 Age of project and Workload of project 17

Chapter 5 Discussion 21

Chapter 6 Conclusion and future work 21

Reference 22

Appendix A 1

First appendix 1

A.1 Useful Links 1

A.2 My Code 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 Xuhui 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 11230 projects, 33347937 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 paper is to replicate the correlation study between project characteristics and pull request merge success in the nine papers mentioned in the article by Zhang Xuhui et al. on project characteristics, and to build on their work to find one or more methods of data analysis to correlate parts of the project characteristics in the dataset. This led to the identification of features in the project characteristics that have a greater impact on pull request merging.

The structure of the paper is as follows: Chapter 2 provides a background to the paper, a basic explanation of pull requests, and a summary of the methods and conclusions of the related papers on the project features to be studied. Chapter 3 explains the characteristics of the projects in the dataset and the partitioning and cleaning of the dataset. The tools and methods used in the analysis of the data are also explained to make it easier for others to copy the code when viewing the paper. Chapter 4 presents the results of the data analysis of the dataset and discusses them. Chapter 5 concludes with a summary of the whole project and a reflection on the results of future work.

# Background & Related Work

## Pull Request

As described in the introduction, pull request is a new distributed development model that works in the following way: 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 the core development team. If the modification is functionally satisfactory, the pull request will be merged; if not, it will be rejected, and the contributor can be asked to make additional changes. The exact process is shown in Figure 1.

图示

描述已自动生成

1. **Pull Request Process**

## Relate work

In recent years, there has been a growing body of research on the impact of project characteristics on the success rate of pull request mergers. In this section, we provide a review of research involving six characteristics of projects to determine how we should replicate their research to investigate the relationship between project characteristics and pull request merge success rates.

• In a study by Soares D M et al. [5] on the impact of programming languages on pull request pairs, they calculated the programming language and 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. In contrast, in the study by Rahman M et al. [6] the number of successful and failed merges of pull requests in different languages was counted by means of a double bar chart and compared to derive the guan of different programming languages in relation to the success rate of merges of pull requests. Whereas in Padhye R et al.'s study [7], they grouped the statistics by language, separately for the number of developers and submissions, and presented as well as analysed them using box-line plots to draw conclusions.

• Similarly, in the study by Rahman M M et al [6], the effect of the number of project bifurcations on the merging of pull requests was investigated, using the same methodology as when studying programming languages, and the effect of bifurcations on merging was also examined through a graphical statistical approach. In a study by Gousios G et al [8], the use of watcher count and star count was proposed to represent the popularity of projects. In the study by Khadke, N et al [2], their main work was to predict the merge outcome of pull requests by using regression models to make predictions using several regression models, where the features they used were the number of forks and the number of watchers. Although their aim is different from ours, when they use regression models for prediction, they can also get the correlation coefficient between the number of forks, the number of viewers and the merge success rate, so that they know whether there is a positive or negative correlation. In contrast, in the study by Tsay J et al [9], the choice was made to use the number of stars to represent the popularity of the project. They used a multi-level mixed effects logistic regression model to predict the correlation between the number of stars and the success rate of the merger, and to find the relationship between them.

•In the studies by Tsay J et al. [9] and Yu Y et al. [10] they both used a multi-level mixed effects logistic regression model to predict the likelihood of accepting a pull request. The effect of project age on the success rate of pull request merging was derived by predicting the correlation coefficient and correlation in the model for the age of the project.

•Similarly, in the study of the effect of project workload on the success rate of pull request mergers, Yu Y et al. [10] used a multi-level mixed effects logistic regression model for correlation analysis, as did project age. In contrast to the above, in Baysal O et al. [11] they first used Kolmogorov-Smirnov tests to examine the distribution of the data and then used Kruskal-Wallis ANOVA to examine the correlation between project workload and pull request.

•For both project activity and project openness, which were used as essential characteristics in the study by Khadke, N. et al. [2] logistic regression models were used to predict the merge success rate of pull requests. The correlation coefficients of these two characteristics on the merge success rate were obtained in the prediction process to derive the correlation with the merge success rate of pull request.

Taken together, these studies demonstrate that pull request acceptance and project characteristics are correlated. However, the above studies do not summarize all the correlations between the characteristics, so the next step in this study is to bring them all together and draw specific conclusions using data analysis.

# Tools and Methodology

## Dataset

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

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Project | Feature | Pull Request |
| Total | 11230 | 96 | 3347937 |

1. **Dataset properties**

From the table we can see that the dataset contains a total of 3347937 pull requests, which come from a total of 11230 projects. And each pull request has 96 features. These 96 indicators are divided into three main categories in the study by X. F. Zhang et al. I focus on the project characteristics, which contain a total of seven indicators from the dataset, grouped into six project characteristic attributes. Table 2 shows the features included in the project characteristics and the features that need to be used to combine with the project characteristics.

|  |  |  |
| --- | --- | --- |
| **Project Characteristics** | | |
| Project characteristics | Feature | Description |
| **Programming language** | *language* | Languages used in the project [4,5,6,7] |
| **Popularity of Project** | *fork\_num* | Number of branches that existed in the project at the time of the pull request [2,4,6,9] |
| **Age of Project** | *project\_age* | Time interval between pull request creation and project creation [4,9,10] |
| **Workload of Project** | *open\_pr\_num* | The number of pull requests that were being reviewed at the time the pull request was uploaded. [4,10,11] |
| **Activeness of Project** | *pushed\_delta* | The time interval between the two most recent pull request requests opened. [2,4] |
| **Openness of Project** | *open\_issue\_num* | Number of issues already open when the pull request is submitted [2,4] |
| **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 [2,4] |

1. **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  open\_issue\_num | 86.4924  251.1323 | 314.88449  754.88578 | 0  0 | 13 | 10936  7263 | 3347937 |
| 32 | 3347937 |
| fork\_num | 768.346 | 2231.8421 | 0 | 78 | 34664 | 3347937 |
| merged\_or\_not | 0.809 | 0.393 | 0 | 1 | 1 | 3347937 |

1. **Summary of our dataset before removing outliers**

**Data Processing**

1. After counting the values of each feature from the dataset we can see that there are outliers in project\_age and the smallest value found is -22, which is not possible. According to further statistics, it is found that there are 341 values in project\_age that are less than 0. Here our treatment is to delete these 341 data items when studying project age, because the total data The total data set is very large and deleting these 341 data items will have no effect on the results.
2. Next, we can see that there are missing values in pushed\_delta and pr\_succ\_rate, where we have chosen to fill in the missing data using the mean value.

## Data analysis

In this section, we will first test the normality of the data using the KS test [19] and we find that for all features in addition to programming languages, *p-value* < 0.05, indicating that the data for each feature is not normally distributed. We then ran chi-square tests for language, *fork\_num, project*\_age and *merged\_or\_not*. A *p-value* < 0.05 was found, indicating that these three items were correlated with whether the pull request was merged or not. The reason why the cardinality test was not conducted for *project\_work, project\_openness* and *project\_activit*y is because all three would use a regression model where the correlation coefficients of the three can be seen and the correlation can be known. Below is a separate analysis of each of the project characteristics individually.

### Programming languages

Six programming languages are included in the dataset. JavaScript, Java, Python, Ruby, Go and Scala, with different projects using different 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. accounting for almost 80 per cent of the projects.

图表, 条形图

描述已自动生成图表, 饼图

描述已自动生成

1. **Programming language count and Percentage pie chart**

### 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* [2,4,6,9] 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.

形状, 矩形

描述已自动生成

(a)

形状

描述已自动生成

(b)

1. **(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**

### 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* [4,9,10].

**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.

图表, 条形图, 直方图

描述已自动生成

1. **Age distribution of projects when pull requests are submitted**

### 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* [4,10,11] in the dataset metrics. A count of this feature in the dataset is performed to see the distribution of this data.

**Descriptive Statistics**

形状, 矩形

描述已自动生成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)

形状, 矩形

描述已自动生成

(b)

1. **(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.**

### Activeness of project

The activeness of a projec is 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 *pushed\_delta* [2,4] in the dataset metrics.

### Openness of a project

The openness of the project is the number of issues that exist at the time of the pull request submission. The higher the number of issues, the higher the openness. The *open\_issue\_num* [2,4] is used in the dataset metrics to indicate this.

### 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]. where *merged\_or\_not* is used to determine whether the pull request is merged, and core\_member is used to determine whether the member submitting the pull pr\_succ\_rate is the acceptance rate of the project when a pull request is submitted.

## Regression model

In this section the models and tests used in the analysis of the data are described.

### Logistic Regression model

In the course of the research analysis, we need to use a regression model to make a prediction of the outcome of the features in the dataset, as this is a dichotomous problem, so a logistic regression model is used to make the prediction. The aim is to see the correlation between the features and the combined results. Here, the logistic regression model included in scikit-learn [13](a machine learning library for python) was chosen to be used for the prediction. Where for the logistic regression model, we chose to use two logistic regression models with a l1 penalty term and a l2 penalty term for prediction analysis and comparison.

### Multi-level Mixed Effects Logistic Regression Model

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

## Association Rules

Association Rules is an important technique in data mining that reflects the interdependence and association between one thing and other things and is used to extract valuable relationships between data items from large amounts of data. In this study, we are using multidimensional association rules to analyse the data [5,15]. An association rule is an implication of X → Y, where X is the prior of the association rule and Y is the successor of the association rule. In layman's terms, X is the cause of an event, Y is the effect of the event, and the presence of X implies the presence of Y. This is the association rule. The relevance of association rules is assessed by two main indicators of interest: support and confidence [5,16].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].

where another metric, lift, is introduced to express the correlation between X and Y. The formula for lift is Lift (X → Y) = Conf (X → Y)/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.

# Results

## Programming languages

For programming languages, two methods have been used to carry out the analysis. The first method is based on the methodology of Rahman M et al. [6] and uses two bar charts to count the number of successful and failed merges of pull requests for different programming language projects respectively. The statistical results are shown in Table 4.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

1. **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.

图表, 条形图

描述已自动生成

1. **Project programming language pull request merge statistics chart**

It can be seen that 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.

图表, 条形图

描述已自动生成

1. **Lift of the rule: language→ merge**

The article by Padhye R et al. [7] is not replicated in this study because in their article 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 the pull request was submitted by a core person, so it is not possible to replicate their article.

## 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 in order to better investigate the relationship between item popularity 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 |

1. **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.

图表, 条形图

描述已自动生成

1. **Popularity of project pull request merge statistics chart**

In the study by Rahman M et al, 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 next subsection as 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, but our dataset does not contain this one feature. Therefore it is not possible to replicate their paper.

## 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 the l2 penalty term.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

1. **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 |

1. **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. Also the failure rate increases as the time interval between the last pull request of the project increases. 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.

## 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 |

1. **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 |

1. **Workload of project merge success or failure statistics table**

From Table 8 we 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 less and less 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 also 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 problem 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.

图表, 条形图

描述已自动生成

1. **Age of project pull request merge statistics chart**

图表

描述已自动生成

1. **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*).

图形用户界面, 表格

描述已自动生成

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 and the project becomes more open, they are using the age of the project as an indicator of project openness whereas 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.

# 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：

* For project development languages, more commonly used and widely available languages such as JAVASCRIPT, PYTHON and GO are more likely to accept pull requests.
* 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, and 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.

# Conclusion and future work

In this study, we used a variety of methods to correlate the factors influencing the acceptance of pull requests, but 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 to go 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.

# Reference

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>

###### 

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**](https://towardsdatascience.com/how-to-run-linear-mixed-effects-models-in-python-jupyter-notebooks-4f8079c4b589%20)

**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**](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>