How to win a star on GitHub

Michel Kraaijeveld
Delft University of Technology
Delft, the Netherlands
Email: j.c.m.kraaijeveld@student.tudelft.nl

Tom den Braber
Delft University of Technology
Delft, the Netherlands
Email: t.d.denbraber@student.tudelft.nl

***Abstract*—**

I. Introduction

More and more people make use of GitHub to host their Open Source Software (OSS)[1]. GitHub is not only a place were people share their software, it has also become a 'social network' for software developers because of its social features. [2] One of those social features is 'starring' a project. Users can show their appreciation for a project by giving it a star. The amount of stars for a project thus gives an indication of how many people appreciate that project.

However, there might be several factors that can be related to the amount of stars of a project. One can think of the functionality that the project offers, but what about projects that offer more or less the same functionality but have a greatly different amount of stars? Other factors such as popularity of the topic of the project, it being easy to use, e.g. a library that can be added, and more, can all have influence.

This paper will be looking at the stars given to OSS projects and whether certain factors are related to the amount of stars a project can possibly get, and if there are key points to focus on when aiming at having the most stars as possible.

II. Problem Description

Although the amount of stars for a project are far from the only indicator of a project's success, they give an indication of how many people actually like the project. For developers, it might be important to know how they can gain the appreciation of other GitHub users: the more people that use or collaborate to a project, the faster it will evolve.

We will now further concretize the subject of this research. The main research question is as follows: 'Can we predict the amount of stars a project on GitHub will get when looking at its characteristics?' This high-level question can be split up into two more concrete questions:

- Which factors influence the popularity of a project?
- To what extent can we accurately predict the amount of stars on a project?

To answer the questions specified earlier, a dataset which contains a lot of data concerning the projects on GitHub is needed. This dataset is available: GHTorrent [3] is a dataset which is constantly mining the GitHub application for more data on projects, and contains all the data that we need for this research.

However, since it also contains data that is not useful for this particular research, we chose to create our own dataset based on GHTorrent. This dataset contains 2000 open source projects with several features defined for each of them and is freely available for anyone to use. It can (not yet) be found on https://github.com/IN4334-group-1/msr.

III. Methodology

Now that the research questions have been discussed, the approach for answering those questions can be explained. We will start by detailing our sampling method. Thereafter, we introduce the features which we think are related to the amount of stars that a repository has. Lastly, we discuss the methods for actually constructing a learning model that we can use to predict the amount of stars for a given project.

A. Sampling Method

The distribution of stars per project appears to be extremely right skewed: there are projects with one or zero stars, while there are much less projects which have a lot of stars, as can be seen in Figure 7. Because of this skewness, the decision was made to use so-called stratified sampling. With stratified sampling, we first have to divide the population into strata. Thereafter, we sample an equal amount of projects from each of these strata; all these samples together form the sample that is being used in the rest of the project.

Each stratum has as characteristic that the amount of stars of the project in that stratum fall within a certain range. The ranges that will be used in this project are: [0, 10], (10, 100], (100, 1000], (1000, +]. Because the dataset is very large, we can also create a quite large sample: out of each stratum, we will randomly select 500 projects for each stratum. The total size of our sample will therefore consists of two thousand projects.

Further, we split up our sample in a training set and a testing set. Out of our sample, we randomly pick half of the projects which will form our training set. The other half of the sample will be used as a testing set.

Although we mentioned that the selecting of projects is completely random, it actually is not. To reduce bias in our sample, we decided to filter out any project by Google, Microsoft or Apple. The way in which they use GitHub is different from most other projects, but they do have a lot of stars, partly due to brand awareness; therefore, they are not included in our sample. Furthermore, deleted projects are also

not contained in the dataset. The reason for this is that not all of the features can be successfully applied to deleted projects, and we don't consider them to hold the key to getting stars on GitHub; as we would expect a successful project to still be around.

B. Features

In order to be able to predict the amount of stars an OSS project will get, a model need to be designed. Therefore a number of features is discussed first, which could be related to the amount of stars on a project. Each of these features fall into one of the high-level features that are specified: general project characteristics, popularity of developers or organisation, and activity. Next to these high-level features, the domain is also considered as that is another aspect that can be related to the amount of stars. After the features are defined, we can use them in combination with a machine learning algorithm to create the actual prediction model.

1) General project characteristics: The first high-level feature contains general features belonging to a project. A feature is considered a general project feature if it cannot be categorized into the other high-level features. In this section the different features and reasons behind choosing them will be explained.

Project country of origin The country of origin refers to the country in which the project was intially created. This can either be based on the developer that created the project, or on the majority of developers in the organisation

Number of commits per developer The total amount of commits that each of the developers in the project have. These commits do not include merge commits.

Number of forks The number of forks is the total amount of forks that the project got since its intial creation. Since forking a repository means that someone is interested and possibly want to make use of the project as a resource for their own developments, it can be a good indication of the amount of stars a project might possibly get.

Number of pull requests This numerical variable contains information about how involved contributors are to the project: do they make a lot of pull requests or not?

Total amount of contributors Someone is considered a contributor when he or she can directly commit to the repository (developer) or when a pull request is merged into the project (external-developer). When a project has more contributors, it might give a prediction on the amount of stars the project will possibly get.

Main programming language The language that is used most throughout the project is considered the main programming language. Since some languages are easier to learn and understand, or is more popular, projects that use that language might be able to attract more users and therefore stars.[4]

How long does the project exists Newer project might have fewer stars, since they are not discovered by the majority of the GitHub users. Therefore the time a project exists, might have influence on the amount of stars the project currently has.

2) Characteristics of developers and organisation: When a developer or organisation is already known on GitHub and has established a group of followers, a new project created by them can get attention from other users more easily. To be able to find out whether this has any relation to the amount of stars a project can possibly get, a couple of features are created to be included in the prediction model.

Average number of followers per developer This is the average number of followers between all developers that contribute to the project.

Maximum number of followers per developer This number indicates the maximum amount of followers the developers that contribute to the project have.

Number of developers in organisation If the project is created by an organisation, this number indicates the amount of developers that belong to the organisation.

Average number of followers per developer in organisation Again, if the project is created by an organisation, this number indicates the average amount of followers the developers in the organisation have.

Maximum number of followers per developer in organisation This feature is like the previous one, except it now shows the maximum amount of followers a developer in the organisation has.

3) Activity: Another thing that we think can have an impact on the amount of stars a project gets, is the activity on that project: is it actively maintained or not?

Number of commits per day This numerical variable measures the average number of commits per day.

Ratio: accepted/total pull requests This numerical variable contains the number of accepted pull requests divided by the total number of pull requests. It is also an indication of how 'open' the project is.

4) Domain: Since some project domains are more popular, it is easier to get stars for projects in that domain, simply because more people are interested in that domain. Therefore the domain is also considered in the model, to be able to make better predictions as features might weight different across domains. Unfortunately, the domain of a project cannot be automatically deduced from the dataset. Therefore, each project in our sample was given a domain manually. A domain is obviously a categorical variable and is based on the 2012 ACM Classification. [5] From this classification, we only considered the main categories. The reasoning behind this is that the amount of subcategories is huge, and if the projects were divided into them there might be domains that would only contain a single project.

C. Finding relations

Now that the features have been listed, we can try to infer relations between those features. More specifically, we have

an independent variable - the amount of stars for a project - and we will try to find relations with the independent variables - the features listed in the previous section.

The machine learning algorithm that first comes to mind is linear regression. However, some of our features are categorical, e.g. the programming language of a project, which makes it impossible to use linear regression. Luckily, there is a variant available which is able to handle categorical variables; this algorithm is called 'multiple linear regression'. Initially, this algorithm will be used. Depending on the results, we will try out other algorithms like random forest and naive bayes.

On top of the initial analysis using multiple linear regression, an attempt will be made to finetune our model so that the prediction is as accurate as possible. The step-wise regression algorithm will be used for this task.

IV. Evaluation

To make sure the created dataset contains the data we expect, there have been numerous validations on it. Whenever the data of one of the features was retrieved, we manually checked some of the entries. This was done picking a number of random samples from the data and looking it up on GitHub to see if the data we have found corresponds with the actual data. Another point that we checked on, were the outliers in the data. Whenever a feature contained entries that were much higher or lower than the other ones, we also manually checked them on GitHub to make sure the outliers are correct and not some anomalies in our data.

A. Threats to validity

When creating the dataset, some issues arose which will be further explained in this section.

1) Domains: The projects in the dataset have been classified into different domains. Since this has been done manually, it can occur that not all projects are classified in the right domains as it introduced subjectiveness. Some projects also span multiple domains and that makes it hard to categorize it correctly. Furthermore, the classification has been done only for the main categories. The projects could have been classified into subcategories, but with a sample size of two thousand projects this would then result in small samples for each of the domains.

2) Validations: The data in the GHTorrent dataset is not fully up to date with the latest changes on GitHub. This made it harder to validate the data that we retrieved, as it differs from the data we find when we look it up on GitHub. An example of this would be the amount of stars a project has. If GHTorrent contains data that is a couple of weeks old, the project might have gained or lost some stars in the meantime.

3) GitHub limitations: A limitation on GitHub's side is that deletions are not contained in the events that GHTorrent uses.[3] This can introduced inconsistent data, as projects could have been deleted without this data being in the GHTorrent dataset.

4) Unknown values: Not all information on GithHub is filled in or can be determined. Examples of this are the country of the developer that is often left blank, or the main language of the project that cannot be determined. These values are saved in our dataset as 'unknown' and often comprise a large portion of the data. Because of that, estimations based on for example countries tend to point towards the unknown value.

V. Results

A. Intermediate Data Exploration

During the intermediate data exploration, the dataset has been fully created and most of the features have been visualized. The only thing that is still missing at this point, is the domain classification as it needs to added manually. Most of the data seems to be strongly right skewed, for example, take a look at Figures 1 - 9. However, there are also plots that did not have this right skew property. These will be explained in more detail, as they require some more understandig of the data.

Countries Figure 13 shows the countries in which the project was created. As can be easily retrieved from the figure, most of the projects tend to be created in the US. However, we also find a large spike for the 'unknown' value, which indicate that a lot of the developers on GitHub have not specified their country of origin.

Programming languages Figure 14 shows the programming languages used in each of the projects in the dataset. Just as with the countries, we see a big spike for the 'unknown' value. The reason behind this is that there are a lot of projects on GitHub that are created, but do not have any commits. It is then not possible for GitHub to define a language for the project, thus we get an entry with 'unknown'.

Forks A project can be either created by a GitHub user, or be forked from an existing project. Because forks tend to behave differently, for example they keep the amount of forks the original project has on GitHub, but do not get the same amount of stars, a histogram was made to see how many of the projects in the dataset are a fork. This histogram can be found in Figure 10 and shows that most of the projects are not a fork.

Project age The age of the projects can be found in Figure 11. The age is defined as the amount of days from initial created, until January 6, 2016. As can be seen from the figure, there are more projects recently created. This is also in line with what you would expect, as GitHub tends to grow each year [1].

Ratio of accepted pull requests Figure 12 shows the percentage of pull requests that was accepted per project. There are two things in this plot that might attract your attention: most of the projects have a percentage of 0 accepted pull request, and there is an increase at 1 (or 100%). Both can be easily explained: a lot of projects

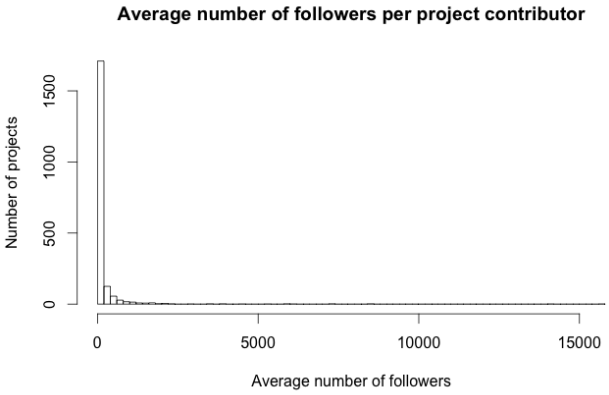

Fig. 1. The average number of followers per project contributor

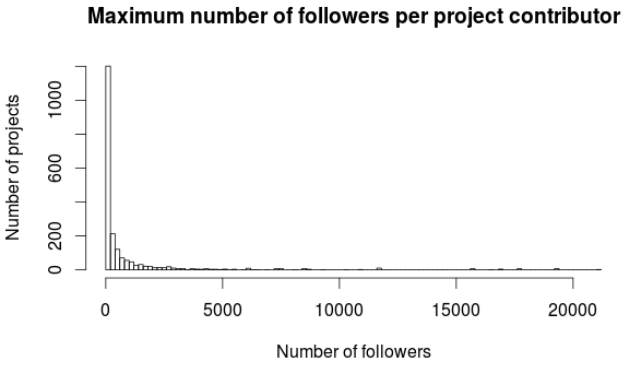

Fig. 2. The maximum number of followers per project contributor

have a small amount of pull requests of which none are accepted, which explaints the spike around 0. On the other hand, a lot of project also have one or two pull requests and have accepted these, which means that they accepted all of them. Since the majority of the projects fall in these two categories, there is also an increase around 100%.

Number of commits per contributor Although Figure 9 looks like a normal right skewed histogram, there is some information missing in this figure. When creating the data for this plot, it was necessary to divide the amount of commits by the amount of contributors. However, there can be so called 'read-only mirrors' on GitHub that have commits without any contributors (for example https://github.com/cran/jomo). When dividing the amount of commits by the amount of developers, this result in a division by 0 which was represented as infinity. These values were not added in the plot.

B. Testing the model

Will be added later on

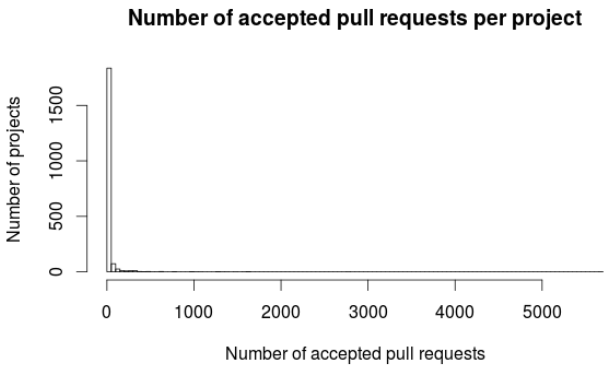

Fig. 3. The number of accepted pull requests per project

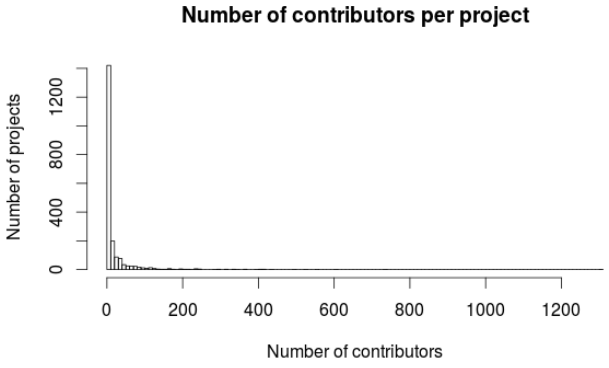

Fig. 4. The number of contributors per project

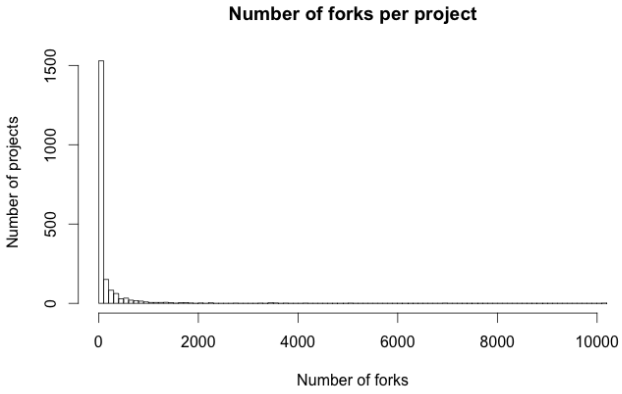

Fig. 5. The number of forks per project

VI. Related Work

Previous work conducted by Chen et al. [6] aimed at predicting the amount of forks for a given repository on GitHub. Our approach was similar to their's, as they had divided their data into different strata in order to get a good sample space. Furthermore, they also had multiple features to create a model to predict the amount of forks. The results

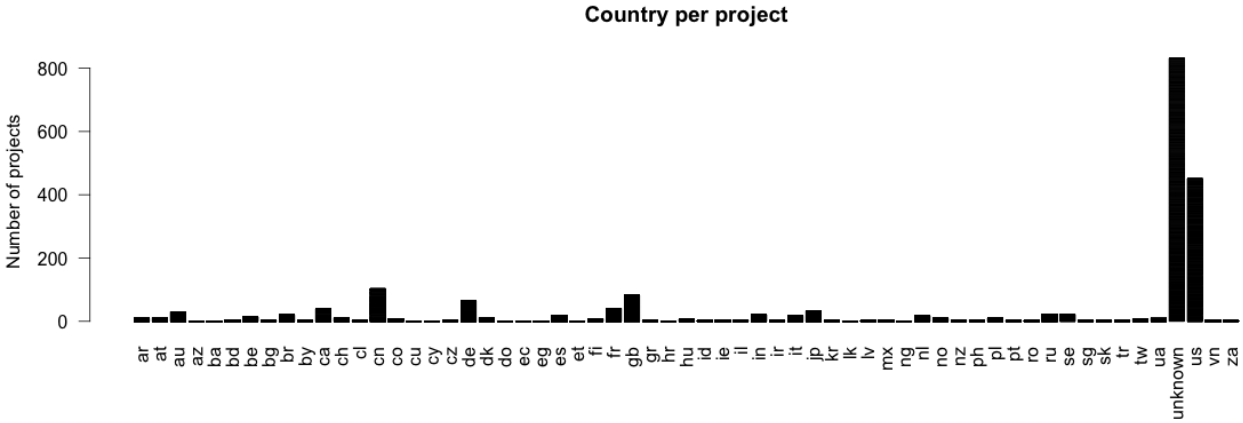

Fig. 13. An overview of the countries where the project was created

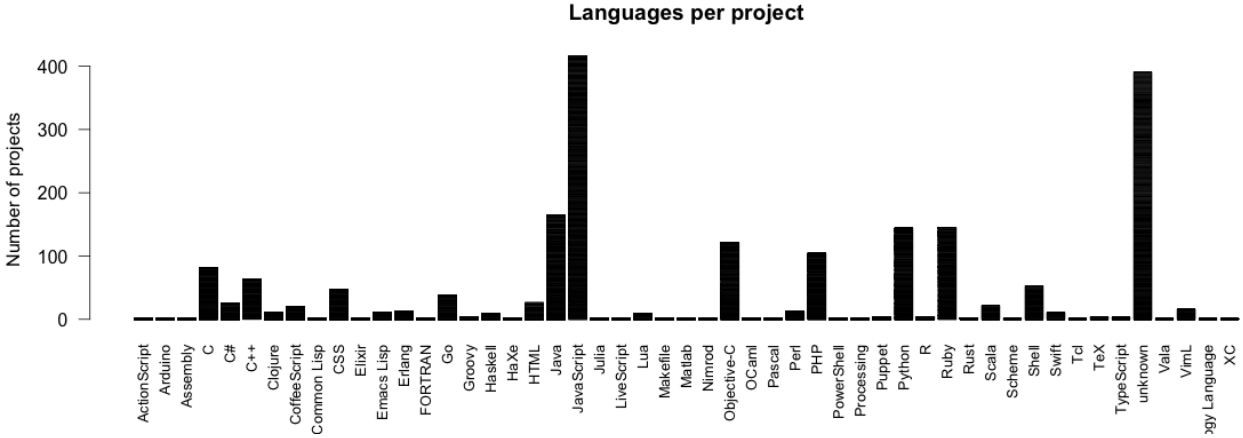

Fig. 14. An overview of the languages user per project

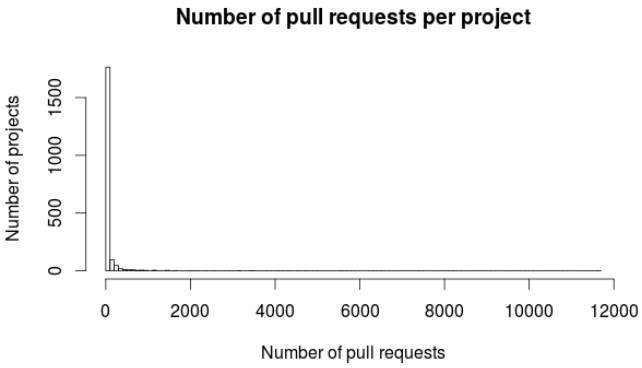

Fig. 6. The number of pull requests per project

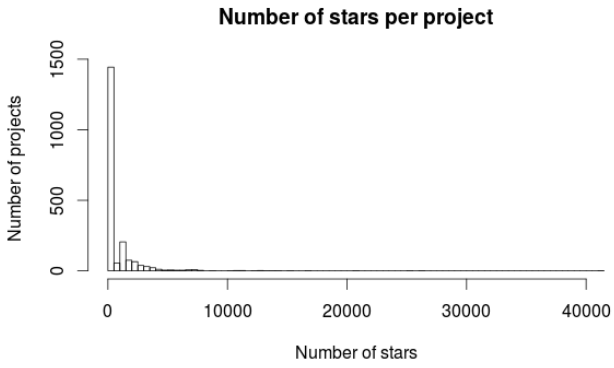

Fig. 7. The number of stars per project

of their research indicated that they could accurately predict the amount of forks for repositories on GitHub and provided valuable insights on the potentials for predicting properties of a GitHub project.

Recent research by Blincoe et al. [7] on popularity of GitHub users, reveals that GitHub users can be influenced by each other to join new projects. They carried out this research

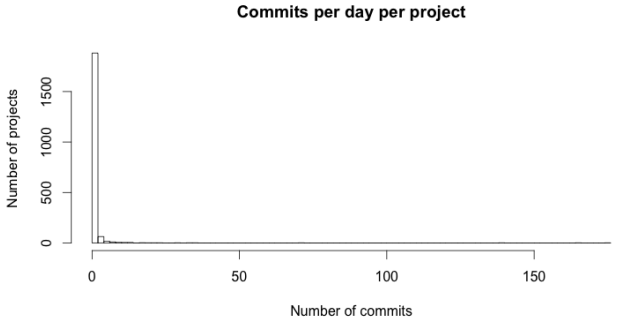

Fig. 8. The number of commits per day per project

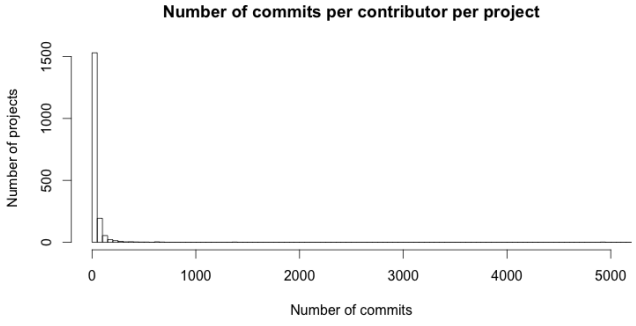

Fig. 9. The number of commits per contributor per project

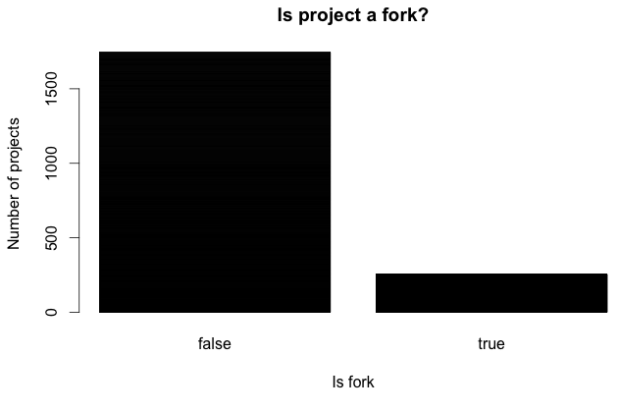

Fig. 10. A histogram showing whether a project is a fork or not

in two ways: by conducting a survey to find out motivations for people to follow others on GitHub and by repository analysis to see the actual influences. Their findings showed that the main reason for following a user on GitHub, is the benefit of having updates and keeping up-to-date with the activities the user participates in. These activities also include joining or discussing new projects, which popular GitHub user attract their followers to.

Research [8] into reported issues for projects on GitHub also provided useful insights. They looked at 100.000 GitHub

Fig. 11. The age of the project in days from January 6, 2016

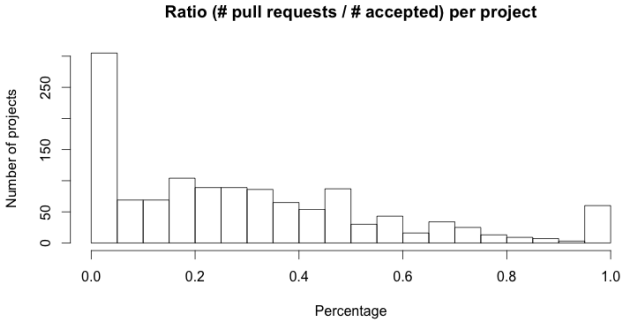

Fig. 12. The ratio of accepted pull requests per project

repositories to see which factors influence the amount of issues that are filed for a project. Their results showed that there is a correlation between the amount of issues and the popularity of the project, where they based popularity on stars - which resulted in a high correlation - or forks - which resulted in a very high correlation.

VII. Future Work

Something that could be related to the amount of stars, that is not considered in this paper, is the documentation of a project. When a project is documented well, or not documented at all, this might have an influence on the appreciation users show for that project. The reason this was not considered in this paper, is because it is hard to measure documentations throughout different projects as some have extensive documentation on their own website, while they are not using the GitHub features for documenting the projected. Furthermore, at the moment of writing it is not possible to check if a project has created a readme file or a wiki page in their repository, as the GHTorrent dataset does not include the files itself. For future research it might therefore be interesting to investigate this feature and include it in a prediction model.

VIII. Conclusion

References

[1] "GitHub 10 million repositories." https://github.com/blog/1724-10-million-repositories. Accessed: January 4, 2016.

[2] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, (New York, NY, USA), pp. 1277–1286, ACM, 2012.

[3] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, (Piscataway, NJ, USA), pp. 233–236, IEEE Press, 2013.

[4] "IEEE the 2015 top ten programming languages." http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages. Accessed: January 4, 2016.

[5] "ACM computing classification system (ccs) 2012." http://dl.acm.org/ccs/ccs.cfm. Accessed: January 4, 2016.

[6] F. Chen, L. Li, J. Jiang, and L. Zhang, "Predicting the number of forks for open source software project," in *Proceedings of the 2014 3rd International Workshop on Evidential Assessment of Software Technologies*, EAST 2014, (New York, NY, USA), pp. 40–47, ACM, 2014.

[7] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on github," *Information and Software Technology*, vol. 70, pp. 30 – 39, 2016.

[8] T. Bissyande, D. Lo, L. Jiang, L. Reveillere, J. Klein, and Y. Le Traon, "Got issues? who cares about it? a large scale investigation of issue trackers from github," in *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pp. 188–197, Nov 2013.