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—**GitHub is a very successful open source software platform on which many projects are hosted. Since stars are included in the rankings on GitHub, they can be a good aspects to focus on when trying to stand out with a project. In this paper, a new dataset is presented based on GitHub data which consists of two thousand projects and multiple features. The dataset is created by using the GHTorrent project and making manual adjustments, such as adding domains to each of the projects. The paper also contains a model - that is used in combination with the created dataset - to predict the amount of stars for an open source software project. The model was applied by using multiple learning algorithms to find out whether prediction of stars is possible and to what extent. Experiments show that the model is able to predict the amount of stars with a high R^2 value, depending on the learning algorithm used. In combination with Random Forest and a log-scaled dataset, the model was able to obtain an R^2 value of 0.88. It shows that the predicted amount of stars is close to the actual number of stars.**

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 and many of GitHub's own rankings are also based on the amount of stars a project has.[3] Therefore one can state that the amount of stars gives an indication of how many people appreciate the 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. Those keypoints might be characteristics they can influence; maybe attracting a popular developer to your project helps a lot. However, another project characteristic that might be related to the number of stars is the number of forks. Unfortunately, that is not a feature that can be influenced by a project owner.

This research will be of help for developers who want to gain more stars for their projects. On top of that, a dataset with interesting features concerning two thousand projects hosted at GitHub can be of interest for other researchers in the mining software repositories field.

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. From a developer perspective, this means that the amount of stars can have multiple effects and advantages. First of all, it is always good to get feedback from users. If the users can indirectly show the developer to keep up the good work by simply starring the project, the developer will notice that his work is appreciated and is something that people like to see or use. Furthermore, when more people appreciate a project, there is a greater chance that they will not only use it, but even collaborate to the project. When more users give input to the developers, this can lead to more bugs being found or features being added, resulting in an overall better project. For a developer it might therefore be useful to get as many stars as possible, as it can help evolve a project faster. Finally, as stated earlier, GitHub has its own ranking systems that mostly involve the amount of stars for a project. If a developer has enough stars in a specific timespan, for instance a day, its project can show up in one of these rankings and will be available to a bigger audience of potential users.

In other words, by getting as many stars on GitHub as possible, there is a possibility that your project will attract a bigger audience and will deliver a better product due to feedback.

As shown, there are a lot of reasons why it is useful to get your project starred on GitHub. However, this paper also aims at finding out which factors might have an influence on the starring-behaviour of users. In order to do that, a dataset is needed which contains information from GitHub. There are at the time of writing three possible ways to get access to such a dataset: through the GitHub API [4], GitHubArchive project [5] or GHTorrent project [6]. The GitHub API can be queried for specific events. However, the main downside is that only the last 90 days are included in the API. The result of this is that important data might be missed, as it might not be possible to retrieve the needed data. An example could be the amount of contributors, as this is determined based on whether a user has made a pull request to a specific project. If it is only possible to go back in time for a very limited period, there is a great chance that some contributors will be missed. Also, the timeframe is not the only limitiation, as there is a limit to 300 events in the API. This means that data could go missing when querying the API at two different times, as the limit of 300 events dropped the entry while it is still less than 90 days old.

GitHubArchive uses the GitHub API to monitor over 20 events [7] that it provides. They have been doing this since the beginning of 2011, but their documentation is lacking a lot of information. Because of that, it is not entirely clear what data there is ready to use and how it is related to each other.

GHTorrent on the other hand also contains GitHub data, but it only contains data from 2012 onwards. However, their documentation is much better and gives a clear overview of the available data and how it can be queried. Another advantage is the online MySQL environment [8] that is available to query their database. The GHTorrent dataset is very large, which makes it a good representation of how GitHub is used in reality and is therefore a good source to sample from.

Upon closer inspection of the GHTorrent dataset, it turns out that it also contains information that is not useful regarding the scope of this paper. Example of this are the repository milestones and issue tracking events. Because this research does not include these events as features, it is not useful to include them in the dataset on which the calculations will be done. Therefore a new dataset is created which only includes the useable events. The data that is included in the new dataset is more extensively explained in Section III, where also the features are explained. The dataset contains two thousand projects and can be found at https://github.com/IN4334-group-1/msr.

With the use of the new dataset, the following question will be answered in this paper: ‘Can we predict the amount of stars a project on GitHub will get by looking at its characteristics?’ This high-level question can be split up into two more concrete questions:

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

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 1. Because of this skewness, the decision was made to use *stratified sampling*[9]. 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, +]. The reasoning behind these ranges is based on the research by Chen et al. where they used the same ones for splitting their data that was based on forks; as the same characteristics applied to the amount of stars. [10] 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.

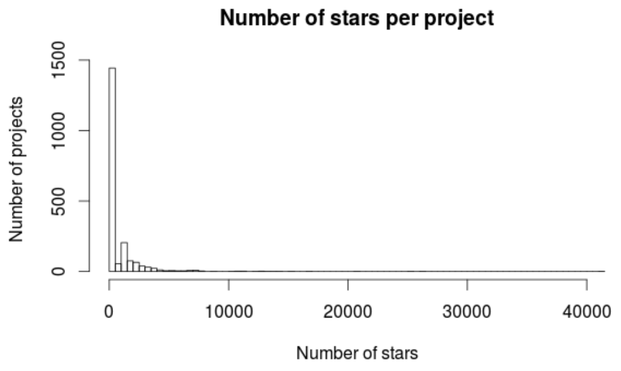


Fig. 1: The number of stars per project

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 (for example, by requiring a contributor to sign a legal agreement [11]), 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

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.

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.[12]

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.

Description available GitHub offers the possibility of giving a one-line description, which is placed directly below the title. Having a good or catchy description might attract users to examine the project further.

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 find out whether this has any relation to the amount of stars a project can possibly get, five 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. Else, the number of developers in the organisation is set to be one.

Average number of followers per developer in organisation If the project is created by an organisation, this number indicates the average amount of followers the developers in the organisation have. Else, the amount of followers of the project owner is inserted here.

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. When the project is owned by a user, the amount of followers of the project owner is inserted here.

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 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 a categorical variable and is based on the 2012 ACM Classification. [13] From this classification, we only considered the main categories. The reasoning behind this is that the amount of subcategories is quite large, 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.

On beforehand, we obviously do not know how and if the data is related. Luckily, there are many learning algorithms available which can be used. These algorithms take a model, which consists of a dependent variable and a list of features. The learning algorithm than takes this model and a set of data and tries to find a useful relation between the dependent variable and the independent variables.

To find the relation between the features and the dependent variable, we will execute the following procedure for each of the algorithms that will be tried.

- Execute 10 times:
  - Split the dataset in a training and a testing set, both consisting of half of the data.
  - Train the model using the chosen learning algorithm and the training data.
  - Use the model to predict the amount of stars for the entries in the testing set.
  - Compare the predictions and the actual values, and calculate the R^2 value.
- Compute the mean of all the R^2 values.

After this procedure, we have a reasonable estimation of how well the learning algorithm handles our dataset.

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 by picking ten random samples from the data and looking them 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. Some examples of this included a wiki with a single contributor (a bot) and over 100,000 commits and read-only repositories without contributors.

A. Threats to validity

When creating the dataset, some issues arose which will be further explained in this section. Also the steps taken to mitagate the threats, where possible, are described.

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. To be able to validate GHTorrent data even though it was not identical to the data currently on GitHub, we introduced a margin for the amounts that we can check. This means that if a certain user was listed to have x followers in GHTorrent, the user should have a number of followers in the range of $[x - 0.05 \times x, x + 0.05 \times x]$ on GitHub. This approach was also applied to the other features, such as stars and commits.

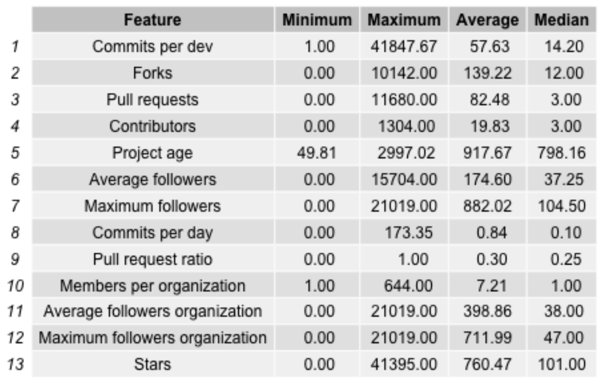
3) GitHub limitations: A limitation on GitHub's side is that deletions are not contained in the events that GHTorrent uses.[14] This can introduce inconsistent data, as projects could have been deleted without this data being in the GHTorrent dataset.

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. Data Exploration

During the data exploration, the dataset has been fully created and all of the features have been visualized. Most of the features are extremely right skewed; these have not been displayed graphically but can be found in Figure 2. This table displays the minimum, maximum, mean and median value for each of the features. It can be easily seen that the average is always higher than the median, which is an indication that the data is right skewed. However, other features did give more interesting insights; this involves mostly the categorical variables. This report does include a visualisation of those features, as well as an explanation as they might require some deeper understanding of the data.



	Feature	Minimum	Maximum	Average	Median
1	Commits per dev	1.00	41847.67	57.63	14.20
2	Forks	0.00	10142.00	139.22	12.00
3	Pull requests	0.00	11680.00	82.48	3.00
4	Contributors	0.00	1304.00	19.83	3.00
5	Project age	49.81	2997.02	917.67	798.16
6	Average followers	0.00	15704.00	174.60	37.25
7	Maximum followers	0.00	21019.00	882.02	104.50
8	Commits per day	0.00	173.35	0.84	0.10
9	Pull request ratio	0.00	1.00	0.30	0.25
10	Members per organization	1.00	644.00	7.21	1.00
11	Average followers organization	0.00	21019.00	398.86	38.00
12	Maximum followers organization	0.00	21019.00	711.99	47.00
13	Stars	0.00	41395.00	760.47	101.00

Fig. 2: The minimum value, maximum value, mean and median of all the right skewed variables

Countries Figure 8 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 9 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 4 and shows that most of the projects are not a fork.

Project age The age of the projects can be found in Figure 5. 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 6 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 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 3 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.

Domains Figure 7 shows the domains and the amount of projects that were classified into that domain. This classification was done manually and is based on the top level domains of the 2012 ACM classification [13]. In addition, two other domains were added - *empty* and *not found* - which respectively indicate that a certain project did not contain any files and that a project could not be found anymore on GitHub.

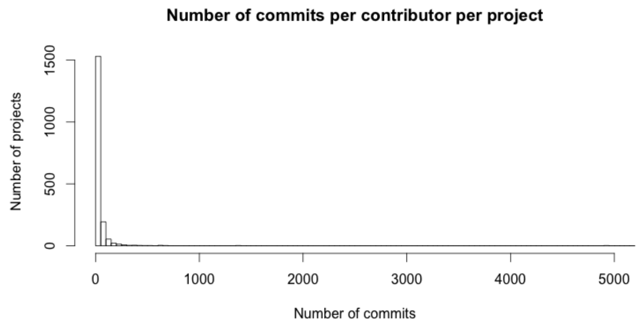


Fig. 3: The number of commits per contributor per project

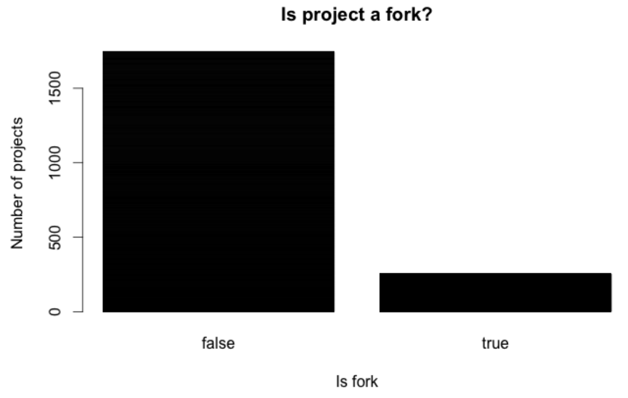


Fig. 4: A histogram showing whether a project is a fork or not

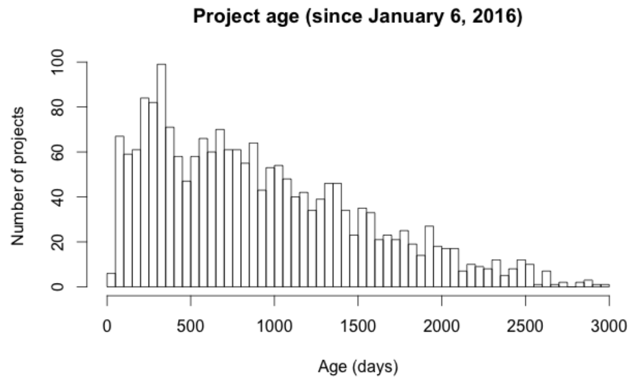


Fig. 5: The age of the project in days from January 6, 2016

B. Relations

Better understanding of the different features was achieved by plotting the correlations. The result of using Spearman's

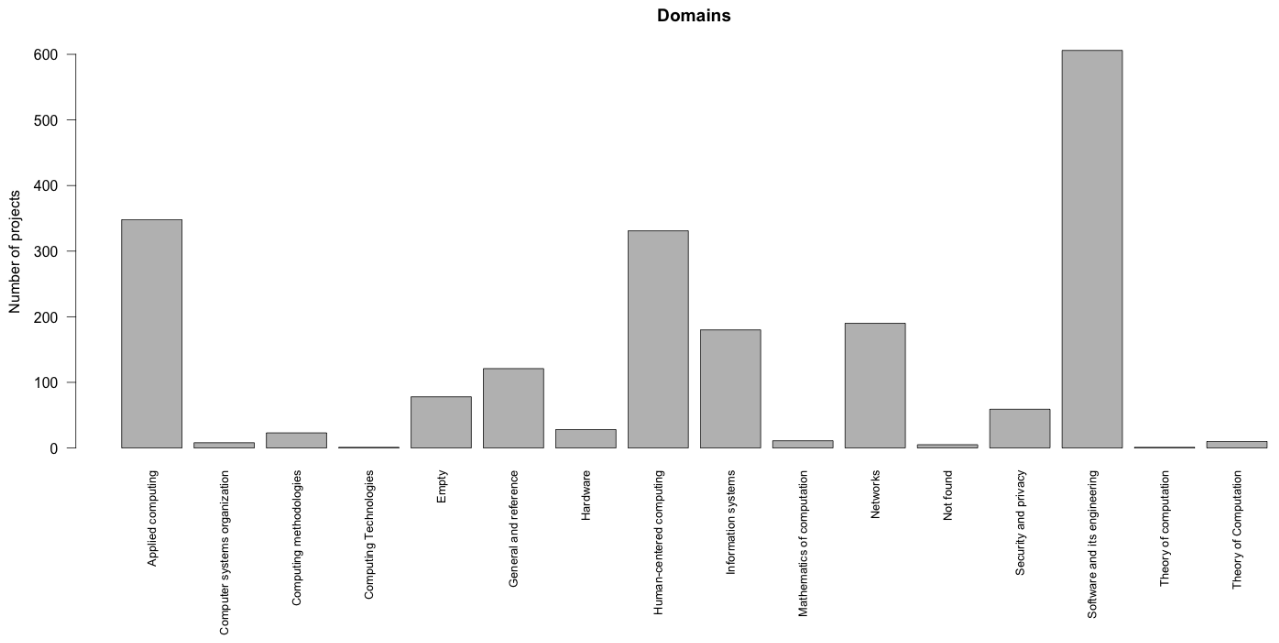


Fig. 7: An overview of the amount of projects per domain

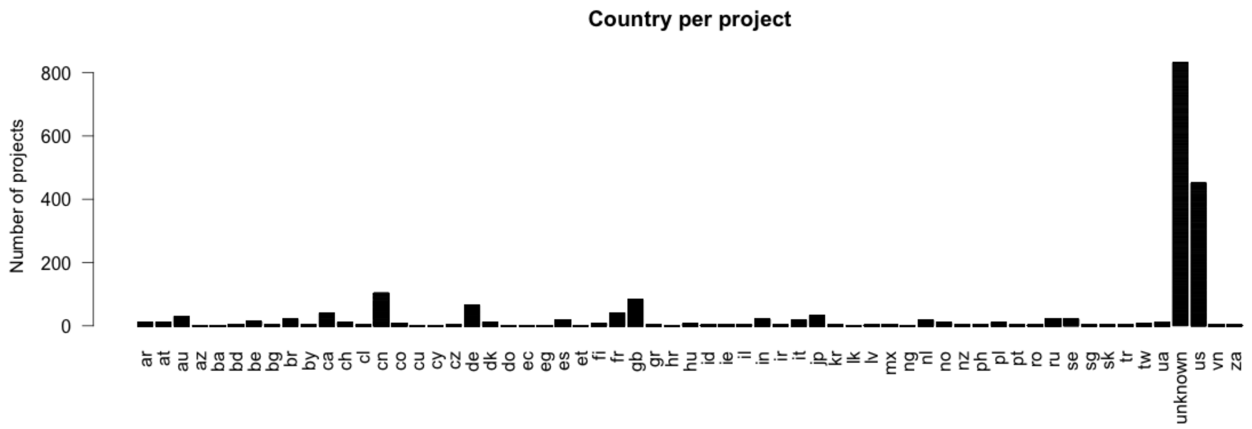


Fig. 8: An overview of the countries where the project was created

rank correlation coefficient [15] on the sampled data can be found in Figure 10. By examining the created plot, some interesting relations have been found. The first one was the negative correlation between the age of a project and its id. This was not very surprising, as the ids of the older projects are lower as these were added to the GHTorrent dataset first. Another notable thing is that the average followers per organization and maximum followers per organization - and the same applies to average followers and maximum followers of contributors - were strongly correlated. Lastly the number of pull request seems to be strongly correlated to the amount of contributors. Again this is not surprising, as a contributor is someone that contributed to the project. Since this can only be done by using a pull request - if the involved developer does not belong to the core developers - the amount of pull requests is influenced by the amount of people that have contributed to a project.

C. Testing the model

After sufficient insight in the features and the relations between the features was gained, a model was created. This model consists of the dependent variable, stars, and has almost all the features listed earlier as features. Based on the exploration, it was decided to drop three variables: the maximum number of followers per organization, the average number of followers per contributor and the pull request acceptance ratio. As stated earlier, the features concerning followers were closely related, so it was decided to keep half

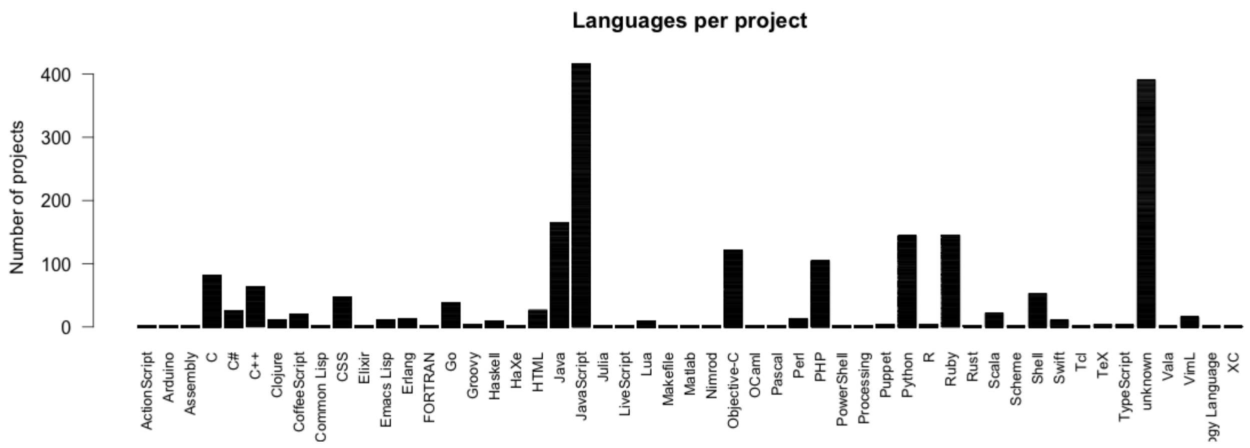


Fig. 9: An overview of the languages user per project

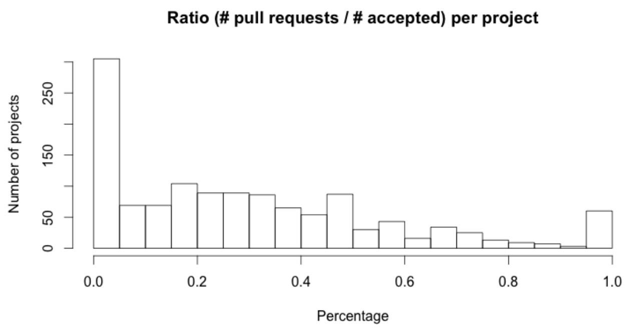


Fig. 6: The ratio of accepted pull requests per project

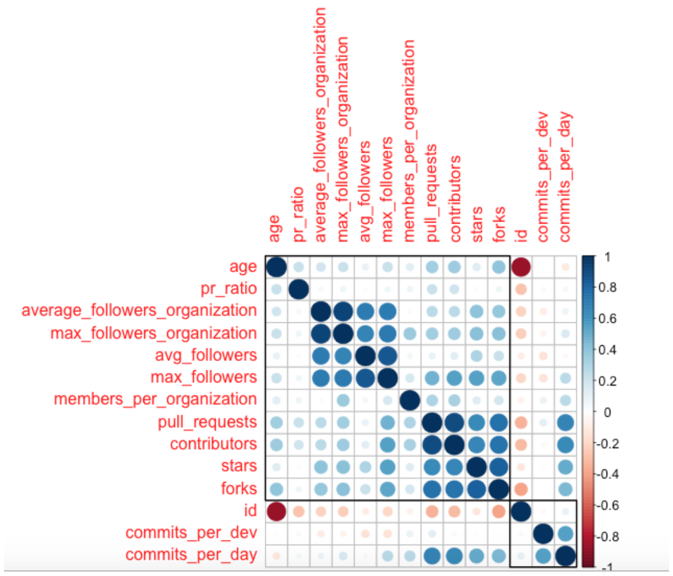


Fig. 10: Correlation between the different features

of them. The pull request acceptance ratio was dropped because of the many repositories that did not had any pull requests at all. This made a lot of entries in our feature table not usable, as it lacked information. To overcome this problem, the pull request acceptance ratio was removed from the model.

Several algorithms have been used to train the model. The results can be found in table I. All the algorithms were run on several versions of the dataset. The first version contained just the values that were retrieved from the GHTorrent dataset, and is indicated by ‘normal’. In the ‘scaled’ version, all numerical features are scaled such that the mean is equal to zero and the standard deviation is equal to one. The last version is the ‘log-scaled’ version. Again each of the numerical features was replaced, but this time by a value that was retrieved from taking the natural logarithm of each value. However, due to the fact that $log(0)$ is undefined and almost all of the numerical features contained entries with zeroes, all of the values were first increased by 1. Each ‘N/A’ entry in the table means that the algorithm could not be run on that dataset, or that it did not yield useful result. Two algorithms that have been tried are not shown in the table; both Multiple Linear Regression and General Linearized Model did not provide useful results on any of the versions of the datasets.

It is interesting to see that 3 out of 4 algorithms performed best on the ‘log-scaled’ version of the dataset. A possible explanation is that most of the chosen algorithms may perform better when there are no negative values involved. The scaled version of the dataset does contain negative values, where the logarithmically scaled version does not, because each value was first increased by one.

The best R^2 was achieved by the Random Forest algorithm on the logarithmically scaled dataset, with $R^2 = 0.88$. This means that 88% of all the variance between the predictions made by the model and the actual dataset can be explained, and is due to variance in the dataset itself; the other 12% cannot be explained and is caused by the fact that the model is not perfect and cannot match the reality exactly. The reason that Random Forest [16] works well with the data, is because of majority voting and feature bagging. The algorithm creates multiple classifications trees based on random subsets of the features and each of these trees counts as a 'vote'. In the end, the votes are counted to make a final classification. Since the dataset contains features that are more heavily correlated with the amount of stars, the combination of majority voting and featuring bagging extracts the important features to create an accurate prediction.

Algorithm	normal	scaled	log-scaled
PCR [1]	0.55	0.63	0.87
PLSR [1]	0.52	0.47	0.86
Stepwise Linear Regression[3]	0.53	0.58	0.05
Neural Networks[4]	N/A	0.24	N/A
Random Forest[5][6]	0.57	0.58	**0.88**

TABLE I: Mean r-squared values for several algorithms used on different versions of the dataset

It was also tried to extract all the items with a certain domain, e.g. 'Networks' and run the various algorithms on a dataset containing just objects with just that domain. However, this did not yield any valuable results. The best R^2 value for those tests was 0.39, which means that still 61% of the variance between the model and the actual data could not be explained.

D. Important features

As is clear from the results shown earlier, it is possible to predict the number of stars of a project relatively accurately. The follow-up question is: which features from the dataset are most important in this prediction? Table II shows the increase in the Mean Squared Error when a certain feature is removed from the model.

It is clear that the number of forks is very useful, as well as the average number of followers per organization member and the number of pull requests; this also is in agreement with the correlations shown earlier, as these features were relative strongly correlated with the number of stars. Both features concerning followers are also quite influential. The average number of commits per day, which resembles the 'activity' of the developers, is also a useful feature. The lack of usefulness of certain features is a bit unexpected. The biggest revelation is that the domain actually does not influence the prediction very much. The size of the organization that owns a repository is also not very influential.

[1]Implementation can be found at https://cran.r-project.org/web/packages/pls/index.html.

[3]Standard R[17] implementation is used.

[4]Implementation can be found at https://cran.r-project.org/web/packages/nnet/index.html.

[5]Implementation can be found at https://cran.r-project.org/web/packages/randomForest/index.html.

[6]In order to make this algorithm work, the country of origin feature was removed, as the implementation that was used was not able to handle a categorical variable with such a large number of possible values.

Feature	MSE increase (%)
Number of forks	53.9
Average number of followers per organization member	26.6
Number of pull requests	25.2
Average number of commits per day	19.3
Maximum number of followers per contributor	18.9
Number of contributors	16.5
Project age	16.3
Project is a fork	13.0
Language	12.1
Average number of commits per developer	11.4
Project has a description	8.0
Domain	5.3
Number of members of organization	4.3

TABLE II: Increase in MSE when a feature is removed

A possible explanation for the importance of the number of forks is that when users fork a repository, they are also likely to give it a star. The goal of forking a repository is to actually make edits to a repository; it shows a certain emotional attachment, the developer in dispute is committing his/her time to improving the project. When someone is willing to do that, it is likely that they also appreciate the project: why else would they spend their time improving it? The same reasoning can be given for the importance of the number of pull requests. The number of followers is also relatively important; users with a lot of followers attract their followers to be involved in new projects [18], which might then gain more and attention and as a side-effect, more stars.

The unimportance of domains could be an indication that the classification that was done by hand was not done very well; another explanation could be that the classification should be more 'fine-grained'. A third explanation is that the domain actually does not have a strong relation with the number of stars of a project. Currently, only the top-level domains from the ACM classification system are used. It is possible that the domain feature is more useful when more specific domains were used.

VI. Related Work

Previous work conducted by Chen et al. [10] 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 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. [18] on popularity of GitHub users, reveals that GitHub users can be influenced by each other to join new projects. They carried out this research 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 [19] into reported issues for projects on GitHub also provided useful insights. They looked at 100.000 GitHub 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.

Another aspect that could be improved is the sampling of the projects. Although we are confident that the approach using the different strata was solid, projects in our sample might share contributors. This implies that it is possible that the different projects in the sample are not independent of each other. To further improve upon this research, it is important that the projects are fully independent.

VIII. Conclusion

In this paper, a brand new dataset and a model was presented which can be used to predict the number of stars for open source software projects on GitHub. The dataset consists of several features that were modeled with different algorithms, including PCR, PLSR, Stepwise linear regression, Neural Networks and Random Forest. The models were trained on half of the dataset and were tested on the remaining half for correctness. Since the data was very skewed, the models were applied to the normal dataset, a scaled one and a log-scaled version. Experiments showed that the different models had varying results, ranging from an R^2 value of 0.05 for stepwise linear regression to a value of 0.87 for PCR and 0.88 for random forest on a log-scaled version of the dataset. This indicates that the amount of stars for a project can be predicted reasonably accurate with the models that were proposed in this paper. Further experiments showed that the number of forks, the number of followers and the number of pull requests were the main features that influenced the error in the prediction.

The provided dataset has a strong potential for providing further interesting insights into characterstics of projects stored on GitHub; it can be obtained at https://github.com/IN4334-group-1/msr.

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] "GitHub about stars." https://help.github.com/articles/about-stars/. Accessed: January 16, 2016.

[4] "GitHub api." https://developer.github.com/v3/. Accessed: January 16, 2016.

[5] "GitHubArchive." https://www.githubarchive.org. Accessed: January 16, 2016.

[6] "GHTorrent." https://ghtorrent.org. Accessed: January 16, 2016.

[7] "GitHub events api." https://developer.github.com/v3/activity/events/. Accessed: January 16, 2016.

[8] "GHTorrent dblite." http://ghtorrent.org/dblite/. Accessed: January 16, 2016.

[9] J. E. Trost, "Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies," *Qualitative Sociology*, vol. 9, no. 1, pp. 54–57, 1986.

[10] 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.

[11] "Google contributing guidelines." https://github.com/google/physical-web/blob/master/CONTRIBUTING.md. Accessed: January 25, 2016.

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

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

[14] 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.

[15] W. W. Daniel, *Spearman rank correlation coefficient*. Applied Nonparametric Statistics 2nd Edition, 1990.

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] "R-project." https://www.r-project.org/. Accessed: January 25, 2016.

[18] 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.

[19] 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.